TOWARDS PRIVACY PROTECTION IN THE ERA OF ADVERSARIAL MACHINE LEARNING: ATTACK AND DEFENSE

HUADI ZHENG

PhD

The Hong Kong Polytechnic University

2021

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

Towards Privacy Protection In the Era of Adversarial Machine Learning: Attack and Defense

Huadi Zheng

A thesis
submitted in partial fulfilment of the requirements
for the degree of

Doctor of Philosophy

January 2021

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

	(Signed)
Huadi Zheng	(Name of student)

Abstract

In recent years, the pervasive application of machine learning has encouraged a boosting amount of artificial intelligent services, such as voice assistant, facial recognition, autonomous driving, word suggestion, and security diagnostics. Essentially, it enables a computer system to learn the underlying patterns from data and represents them in a model, which is then integrated into designated software for assessing new input. While machine learning has significantly reshaped the modern paradigm of system development, it also stirs up extensive social debates over privacy and confidentiality concerns. In particular, adversarial machine learning has grown in importance as researchers discover that a trained model can be deceived, extracted, inverted or applied in malicious inference. Nevertheless, up to now, the understanding of privacy risks and the countermeasures against them remain limited. To unveil privacy challenges and tackle potential vulnerabilities, I focus my PhD study on the emerging attacks and defenses in the context of adversarial machine learning. Adversarial machine learning originally refers to the manipulation of model behavior by supplying deceptive samples. With the rapid development of alternative attacks such as model extraction and membership inference, it has been bestrewed in a broader domain — corruption of functionality and confidentiality with respect to the adoption of machine learning, where new threats are not only presented in the decision stage but also demonstrated across the pipeline of machine learning.

The works described in this thesis are mainly divided into three parts, in a top-down order of attack surfaces, from model prediction to data collection. In the first part, I present a novel mechanism for preventing the extraction of private decision boundary on machine learning services. The proposal consists of obfuscating the output of a classifier with the guarantee of boundary differential privacy, in such a way that fine-grained queries designed to infer the boundary have their accuracy sufficiently diminished in the critical zone, thus hampering the goal of delineating a clear inter-class border. In the second part, I present a side-channel attack system MISSILE to infer sensitive indoor locations in a given premise with machine learning inference. A spyware can stealthily collect these sensory data from typical inertial sensors, such as accelerator, gyroscope and magnetic sensor. In the third part, I will turn to the very source of data collection and study the future of privacy-preserving data collection with an empirical evaluation of local differential privacy and federated learning under a designated task. Finally, I conclude the insights revealed in this study and discuss possible directions of privacy protection with adversarial machine learning in mind.

Acknowledgements

First of all, I would like to express sincerest gratitude to my supervisor, Dr. Haibo Hu, for supporting my research continuously and always looking for the best in me. He offers this exceptional opportunity for me to broaden my horizon and pursue a PhD degree in computer science. I would never imagine going this far in the path of my study without his guidance and instruction. It seems like yesterday this confused undergraduate student was exploring for master's study and received an unexpected chance from PolyU. I owe Dr. Hu my understanding for conducting research properly in the area of security and privacy. His profound visions and critical judgment inspire me to strive for a better self when times are tough and always keep a rigorous mindset in the scientific community.

Secondly, my special thanks are extended to Dr. Qingqing Ye and Dr. Zimu Zheng for many helpful discussions during our collaboration and motivating me to explore challenging research areas. I would also like to thank Dr. Dan Wang and Dr. Ming Li for their kind help. I am honored to be an early member of the ASTAPLE group and would like to express my appreciation for the company of fellow groupmates, Mr. Ziyang Han, Miss Tang Li, Mr. Chun Ho Kong, Dr. Kai Huang, Miss Rong Du, Mr. Yue Fu, Mr. Haotian Yan, Ms. Yaxin Xiao, Mr. Kai Ze, Miss Qiuyu Qian, Mr. Xiao Wang and Dr. Guanghui Zhang. Besides, I would like to thank other colleagues in the lab, Dr. Yin Xiao, Mr. Yonggui Cao, Ms. Lina Zhou, Miss Zilan Pan, Miss Yin Zhang, for bringing joy and sharing the life in this office.

I would also like to thank Wuhua Hall for offering me the chance to become a hall tutor and securing a comfortable place for my residency in Hong Kong. I thank Miss Ruojia Lin, Mr. Kunal Rids, Ms. Yuetyee Yau and Dr. Allen Cheong for their support. My friends, Mr. Qi Zhou, Mr. Ziming Zhang, Miss Ljjia Zuo, Mr. Gaoyuan He, Miss Shilpa Gurung and many others, I sincerely appreciate all kinds of help from you. Particularly, Miss Xiaomin Zhou, thank you for making my world a better place in the darkest times.

Last but not least, I am genuinely grateful to my parents and my better half for supporting my decision to embark on this long journey throughout these years. Being the only son and the oldest grandchildren in the family, I appreciate all the sacrifices that my parents made to ensure that I have a healthy body and receive a quality education.

Long live Hoshino Gen and Michael Buble.

Publications Arising from the Thesis

Huadi Zheng and Haibo Hu. "MISSILE: A System of Mobile Inertial Sensor-Based Sensitive Indoor Location Eavesdropping." IEEE Transactions on Information Forensics and Security (IEEE-TIFS), Volume 15, September 2019, pp. 3137-3151.

Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. "A Boundary Differential Private Layer against Machine Learning Model Extraction Attacks." Proceedings of the 24th European Symposium on Research in Computer Security (ESORICS '19), Luxembourg, September 2019, pp. 66-83.

Qingqing Ye, Haibo Hu, Xiaofeng Meng, and **Huadi Zheng**. "PrivKV: Key-Value Data Collection with Local Differential Privacy." Proceedings of 40th IEEE Symposium on Security and Privacy (SP'19), San Francisco, USA, May 2019, pp 311-325.

Huadi Zheng, Haibo Hu and Ziyang Han. "Preserving User Privacy For Machine Learning: Local Differential Privacy or Federated Machine Learning?" Proceedings of 1st International Workshop on Federated Machine Learning for User Privacy and Data Confidentiality (FML'19), in conjunction with IJCAI'19, Macau, August 2019. [**Best Theory Paper Award**]

Zimu Zheng, Yuqi Wang, Quanyu Dai, **Huadi Zheng**, Dan Wang. "Metadata-driven Task Relation Discovery for Multi-task Learning." Proceedings of the 28th International Joint Conference on Articial Intelligence (IJCAI'19), Macau, August 2019, pp. 4426–4432.

Huadi Zheng, Haibo Hu, and Ziyang Han, "Preserving User Privacy for Machine Learning: Local Differential Privacy or Federated Machine Learning?" IEEE Intelligent Systems (IEEE-IS). 35(4): pp 5-14, 2020.

Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. "Protecting Decision Boundary of Machine Learning Model With Differentially Private Perturbation." IEEE Transactions on Dependable and Secure Computing (IEEE-TDSC), accepted to appear, 2020.

Tianqi Wen, Haibo Hu, **Huadi Zheng**. "An Extraction Attack on Image Recognition Model Using VAE-kdtree Model." International Workshop on Advanced Image Technology (IWAIT'21), Jan 2021. [**Best Paper Award**]

Qingqing Ye, Haibo Hu, Ninghui Li, Xiaofeng Meng, **Huadi Zheng**, Haotian Yan. "Beyond Value Perturbation: Local Differential Privacy in the Temporal Setting." Proceedings of IEEE International Conference on Computer Communications (INFOCOM'21), May 2021, accepted to appear.

Zimu Zheng, **Huadi Zheng**, Dan Wang, Haibo Hu, Abraham Lam Hang-yat, Fu Xiao. "Industry AIOps: Data-driven Chiller Performance Analytics for HVAC Control." Automation in Construction (AutoCon), Under Review.

Qingqing Ye, Haibo Hu, Xiaofeng Meng, **Huadi Zheng**, Chengfang Fang, Jie Shi. "PrivKVM*: Revisiting Key-Value Statistics Estimation with Local Differential Privacy." IEEE Transactions on Dependable and Secure Computing (IEEE-TDSC), Under Review.

Table of Contents

List of Figures

List of Tables

Introduction 1

The breakthroughs in communication and information technology have significantly reformed the modern lifestyle, as we can continue to work from home digitally during the difficult time of a pandemic. Meanwhile, an unprecedented amount of data are generated when we go online shopping with mobile phones, monitor fitness with a smart watch, or comment on social networks. Coupled with the improvement in semiconductor fabrication and parallel computation, massive data motivate the rapid development of artificial intelligence (AI), which is characterized by the pervasive application of machine learning, such as voice assistant, facial recognition, autonomous driving, word suggestion and security diagnostics. In contrast to rule-based software, machine learning enables a computer program to automatically extract patterns (e.g., stochastic gradient descent) from given samples (i.e., training data), and then provide estimation on new unseen samples (i.e., testing data).

Although the amassing of data provides convenience to construct an AI model, it poses a number of threats to individual and corporate freedom over the control of secrecy. On the one hand, data collectors such as government departments and commercial companies can distinguish user's identity, analyze personal behavior, and even implicitly influence one's decision-making process without any consent, leading to the tightened laws against privacy infringement [18], [76]. Besides, even if the user only provides sanitized data (e.g., removal of real data attribute), machine learning significantly reduces the effort to perform side-channel inference and expose original information. On the other hand, given the training process's stochastic nature, the model constructed from user's data is not guaranteed to be secret-free [26], [81]. It can implicitly memorize a sample and later reveal that information under the manipulation of an adversary. Furthermore, model prediction can expose internal formation, which can be extracted for replication without any security breach at the system level. This breaks the confidentiality of intellectual property and often lays the foundation of other privacy-invasive inference attacks, as discussed in the following sections.

1.1 Emerging Privacy Concerns

Privacy is gradually becoming a prerequisite of modern society, which offers seclusion freedom of personal or collective information. The last decade has witnessed both the abuse of private information as well as the significant recognition of privacy all around the world. Particularly, in 2013, a former NSA contractor Edward Snowden revealed how the government organization ran a surveillance program globally by prying into personal activities [88], opening up an intensive global debate about the rights to maintain privacy in the midst of digital revolution. This topic has only expanded as Facebook-Cambridge Analytica data scandal [39] exacerbates user's outpouring rage in the privacy policy. A British consulting firm Cambridge Analytica profiled and harvested personal data from fifty million unwitting Facebook users, which were then provided to a political party for profit. The consulting company eventually filed for bankruptcy after a series of administrative fine. To combat such data abuse, US Federal Trade Commission has filed more than 130 lawsuits against spyware and 50 against general violation of privacy corruption practice [76]; and EU has adopted the more stringent "General Data Protection Regulation" (GDPR) to supercede the "Data Protection Directive" and enforced it in May 2018 [18], which elaborates how data should be collected, retrieved and transferred between different parties. Understanding the importance of privacy concerns not only raises consumer's awareness but also supports the sustainable development of corporation given the increasing scrutiny of regulation.

Unfortunately, with the increasing adoption of sophisticated AI applications, even the best practice of law enforcement can not fully prevent privacy infringement from adversarial machine learning. It has been reported that Amazon smart speaker Echo automatically recorded private discussion and sent the audio to one of the victim's contacts [32]. As it has become apparent in the past several years, new privacy challenges have emerged along with the vulnerability of machine learning and its application in malicious inference. There is an immense demand in the scientific community for advancing privacy protection under the broad vision of adversarial machine learning, as introduced in the following section.

1.2 Adversarial Machine Learning

Despite the momentous advances in AI adoption, our comprehension of the loopholes inherent to such systems and how to protect against them is still in its infancy. Early research in the machine learning community discovers that a well-trained model can be intentionally fooled using deceptive samples, like optical illusions for machines, which sparks heated discussion over a model's

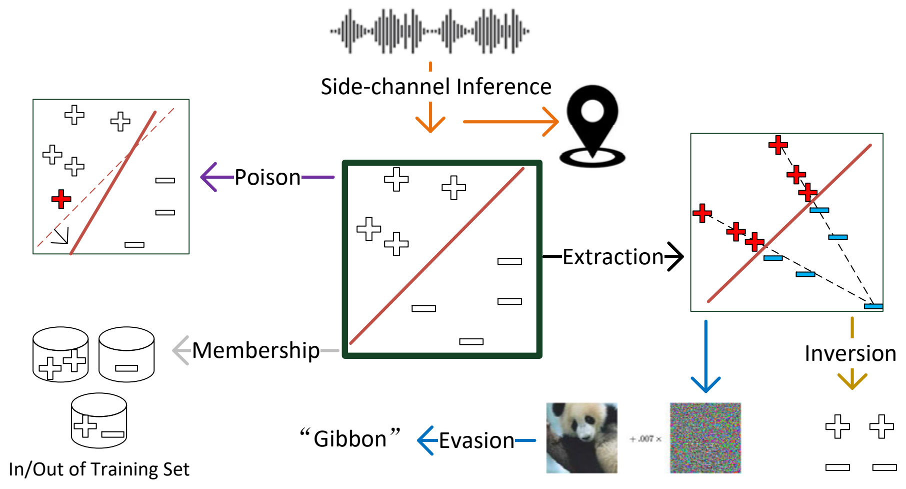


Fig. 1.1: Types of Attacks

inherited vulnerabilities when under adversarial input manipulation. We refer here to adversarial machine learning in a broader sense — corruption of functionality and confidentiality with respect to the adoption of machine learning. This aligns with the rapid development of alternative attacks such as model extraction and membership inference, where new threats are not only presented in the decision stage but also demonstrated across the pipeline of machine learning. We now address the spectrum of adversaries in the taxonomy of adversarial machine learning and our prioritized focus in this thesis. As shown in Fig. 1.1, based on the nature of the attack, adversarial machine learning can be grouped into six categories. The black-frame box at the center represents a normal trained binary model with its decision boundary and the arrows represent feasible malicious attacks. Specifically, the details of each attack are listed below.

- **Model Evasion**: As mentioned before, recent study has shown that model prediction is susceptible to the manipulation of adversarial sample. Such a sample can be efficiently produced using backward algorithms such as fast gradient sign method [28]. Essentially, the effectiveness of machine learning relies on the assumption that training data and testing data belong to an identical distribution, which is usually not the case when an adversary presents. Consequently, this leads to inconsistent prediction over a human-indistinguishable input.
- **Model Inversion**: Turning a model back to its original data seems impossible but it has been achieved by [26]. Recognizable images of faces are inverted from a trained model with only name and prediction confidence provided, which is directly privacy-invasive to the users contributing to the training set.

- **Membership Inference**: Given a data sample and query access to a model, membership inference can discover whether that particular record is presented in training dataset. To some extent, this can be viewed as a weak inversion attack but it still poses the same threats to breach participant identities and business confidentiality. It is achieved by sophisticated shadow model approach [81] and then significantly improved with only a statistical threshold on prediction confidence [78]. We utilize this attack to evaluate the privacy leakage in federated machine learning.
- **Model Extraction**: Researchers have concerningly found that query access can be leveraged to explore model internals and replicate a substitute model with a significant agreement to the original one. Prior work has evolved from extracting basic logistic regression classifier [86] to advanced image recognizer [73]. In particular, model extraction is extremely threatening as it provides a white-box model to the former three attacks and thus becomes one of our prioritized targets in this study.
- **Model Poisoning**: Poisoning a model can either corrupt prediction accuracy (functionality) or backdoor a designated sample (integrity) [40]. This attack takes place during the training stage when malicious training data are supplied through malicious labeling and crowdsourcing.
- **Side-channel Inference**: With the advent of machine learning, unimportant and irrelevant data is no longer insignificant to personal privacy. Efforts required for linkage or pattern mining is largely reduced in inference attacks [67]. To call for broad attention to its privacy-invasive threat, we explore learning-based inference on indoor locations with unprivileged sensor data.

1.3 Contributions and Thesis Organization

The research problems investigated in this thesis can be categorized into three levels of attack surfaces as shown in Fig. 1.2 with two examples of machine learning services. In the model surface (e.g., neural network), we prioritized our focus on model extraction, the stepping stone towards other adversarial machine learning. Countermeasures are proposed using differential privacy. Then in the data pool surface (e.g., collected sensor data), we elaborate the

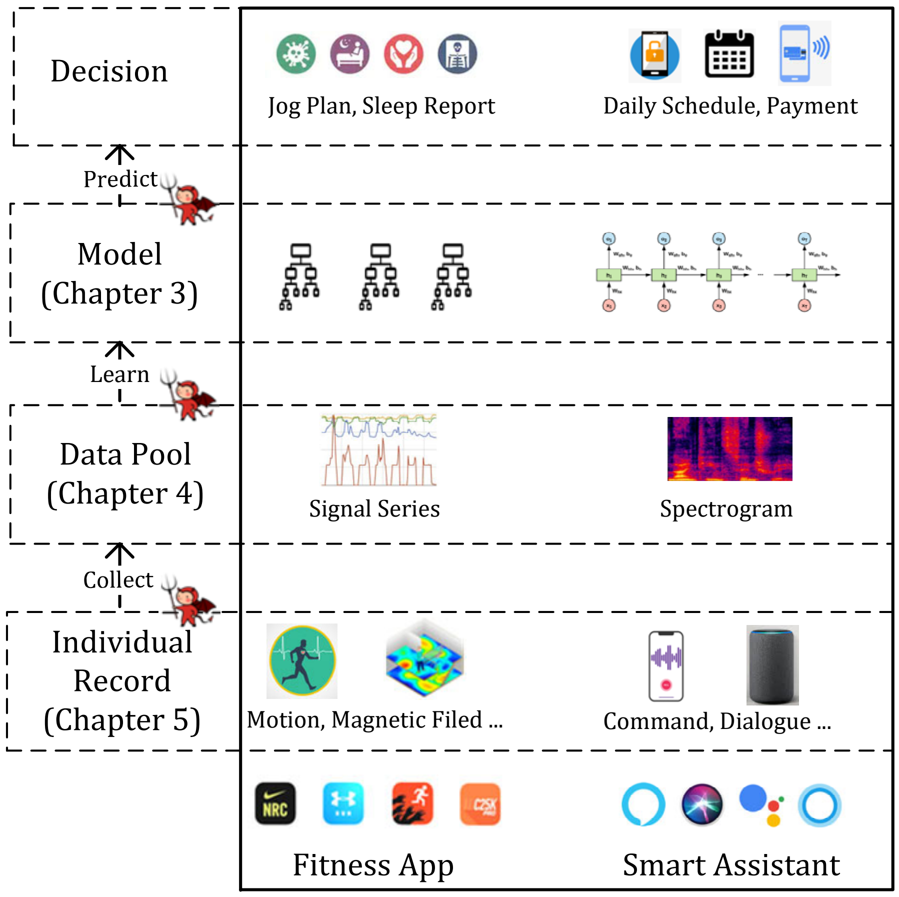


Fig. 1.2: Attack surface

feasibility of a learning-based inference attack on sensor data to comprise private indoor whereabouts. Finally, in the surface of the individual record (e.g., a speech command), where data are about to be collected, we provide an empirical study on privacy-preserving collection under adversarial attacks.

Specifically, this thesis makes the following contributions as presented in each chapter:

- **Chapter 2**: We provide a comprehensive review of related literature extraction attacks and defenses, differential privacy, side-channel inference and localization.
- **Chapter ??**: We elaborate on the basic preliminary concepts applied throughout this thesis.
- **Chapter 3**: For model surface, we present a differentially private countermeasure against model extraction. A novel boundary differentially private layer is proposed against extraction attacks on machine learning services. The proposal offers boundary differential privacy in a

user-specified boundary-sensitive zone by obfuscating the output of a classifier, in such a way that fine-grained queries designed to infer the boundary have their accuracy sufficiently diminished in the critical zone. To identify sensitive queries that fall in a zone, we develop an efficient approach using corner points as indicators. Boundary randomized response is designed as the building block for the perturbation algorithm, followed by a generalization to multiclass model and an adaptive version that can protect a soft margin of the decision boundary. The effectiveness and flexibility of this defense layer on protecting decision boundary is verified through extensive experimental results.

- **Chapter 4**: For data pool surface, we introduce an inference attack that can eavesdrop on user's sensitive indoor whereabouts using unprivileged mobile sensors and machine learning. The key idea is to identify a sensitive indoor location (e.g., an office) using multiple structural characteristics (e.g., turnings in a corridor, pausing of motion to open a fire stop door, or taking an elevator). These characteristics lead to unique patterns in mobile sensor readings and constitute the signature of this location. To evaluate this issue, the thesis proposes the mobile inertial sensor-based sensitive indoor location eavesdropping (MISSILE) system to perform inference using only unprivileged sensors, such as accelerometer, gyroscope and magnetic field sensor. The performance of this inference attack is experimented in our campus with 15 sensitive indoor locations.
- **Chapter 5** For individual record surface, we discuss the future of private data collection against adversarial probing with two distributed data analytical tools, local differential privacy and federated machine learning. The former one is a theoretical privacy notation that can be achieved by different algorithms, while the latter one is a generic distributed learning framework without theoretical provable privacy. Both tools avoid direct access to personal data while still retaining high utility. We conduct a comparative study of both tools to solve a common set of classification problems in mobile scenarios. Important insights into their performance are provided in terms of classification performance, privacy loss, computation and communication cost.
- **Chapter 6** We conclude the outcomes of this thesis and offer new directions for future works in privacy protection.

Preliminary and Literature Review

2

2.1 Supervised Machine Learning Model

A dataset $\mathcal{X}$ contains samples in a d-dimensional feature space. Each sample has a membership in a set of predefined classes called *labels*. Supervised machine learning trains a statistical model by such sample-label pairs to make predictions of labels on unknown samples. In this thesis we focus on classification models which have K possible outputs. Formally, a classification model f produces a response y to a query sample $\boldsymbol{x}$ as follows.

$$y = f(\boldsymbol{x}) = \begin{cases} \text{“class 1” label} \\ \text{“class 2” label} \\ \dots \\ \text{“class K” label} \end{cases} .$$

Classification models have been widely adopted in many machine learning applications, such as activity categorization, face recognition and speaker identification. Depending on the nature of these applications, the model f can be either linear (e.g., logistic regression) or non-linear (e.g., neural network). To train a supervised model, a popular parametric approach is to minimize the empirical risk between the prediction of the model and the real label by adjusting model parameter W as follows.

$$\arg\min_{W} \sum_{x \in \mathcal{X}} \mathcal{L}(f(x, W), y).$$

2.2 Model Extraction

In a model extraction attack, a malicious party attempts to replicate a model from the original one by continuously exploiting the prediction API. Technically any queries can constitute such an attack. However, the more queries the more likely this malicious attack will be exposed. As such, in the literature most model extraction attacks fabricate *fine-tuned queries* by differential techniques such as line search [58], [86] and Jacobian augmentation[74].

These queries are carefully selected to capture the information about decision boundary where prediction results vary drastically.

Formally, a model extraction attack selects a set of fine-tuned queries $\mathcal{X}_{diff}$ and obtains their responses $\mathcal{Y}_{diff}$ to train a replica model f'.

$$\begin{aligned}
&\mathcal{X}_{diff} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}, \quad \boldsymbol{x} \in \mathbb{R}^d, \\
&\mathcal{Y}_{diff} = \{y_1, y_2, \ldots, y_n\}, \quad y \in \mathbb{R}^1, \\
&\exists \boldsymbol{x},\ \boldsymbol{x'} \in \mathcal{X}_{diff},\ dist(\boldsymbol{x}, \boldsymbol{x'}) = \delta\ \wedge\ y \neq y',
\end{aligned}$$

where $dist(\cdot)$[1] measures the distance between two queries and δ is the unit distance adopted in the differential techniques when searching for boundary, i.e., where two corresponding responses $y \neq y'$.

Machine-learning-as-a-service (MLaaS) has furnished model extraction attacks through the rich information available from prediction API. Tramer *et al.* [86] proposed extraction methods that leveraged the confidence information in the API and managed to extract the full set of model parameters using equation-solving. Papernot [74] *et al.* introduced a Jacobian-based data augmentation technique to extend queries and to train a substitute DNN. Similarly, Juuti *et al.* [41] leveraged both optimal hyperparameters and Jacobian-based data augmentation to extract models under their generalized framework. Orekondy *et al.* [71] proposed a knockoff model to steal the functionality of an image classifier and developed reinforce-based query strategy using multi-armed bandits problem. Pal *et al.* [73] further improved the extraction attack on image model without annotated data. Their hybrid strategy combines greedy clustering with adversarial samples. Yu *et al.* [98] proposed new adversarial samples generation technqiue named FeatureFool using feature-based optimization algorithm and performed model extraction on MLaaS platform. Besides extracting model parameters, Wang *et al.* [89] also extracted the hyperparamters of a fully trained model by utilizing the zero gradient technique. Oh *et al.* [69] developed a model-of-model to infer internal information of a neural network such as layer type and kernel sizes.

Model extraction without confidence is similar to *learning with membership query* [4], [87], which learns a concept through querying membership on an oracle. This technique has been exploited by Lowd *et al.* to extract binary

[1] In general, this notation can be any distance metrics (e.g., Manhattan distance, Euclidean distance). The implications of distance metrics to detailed algorithms will be discussed in Chapter 3.2.1.1.

classifiers [58]. They used line search to produce optimized queries for linear model extraction. This technique was extended by Tramer *et al.* [86] to non-linear models such as a polynomial kernel support vector machine. They adopted adaptive techniques such as active learning to synthesize fine-tuned queries and to approximate the decision boundary of a model. Pal *et al.* [73] further improved the extraction attack using only top-1 label on image model.

As for the defense, confidence rounding and ensemble model were shown effective against equation-solving extractions in [86]. Lee *et al.* [52] proposed perturbations using the mechanism of reverse sigmoid to inject deceptive noises to output confidence, which preserved the validity of top and bottom rank labels. Kesarwani *et al.* [46] monitored user-server streams to evaluate the threat level of model extraction with two strategies based on entropy and compact model summaries. The former derived information gain with a decision tree while the latter measured feature coverage of the input space partitioned by source model, both of which were highly correlated to extraction level. Juuti *et al.* [41] adopted a different approach to monitor consecutive queries based on the uniqueness of extraction behavior. A warning would be generated when queries deviated from a benign distribution due to malicious probing. Quiring *et al.* [77] adopted the notion of closeness-to-the-boundary in digital watermarking and applied it to protect against extraction attacks on decision trees. The defense strategy was devised from protection of watermark detector and it monitored the number of queries that fell into security margin. They proposed a remote verification mechanism to determine the model ownership using the watermark implanted in DNN.

2.3 Differential Privacy

Differential privacy (DP) was first proposed by Dwork [24] to guarantee the privacy of a centralized dataset with standardized mathematical notation. It is proposed to bounds data sanitation with a measurable budget so that sensitive information can be released with a strong privacy guarantee. In centralized sanitation, all sensitive data are processed in one place and it is primarily defined in terms of adjacent datasets that differ on one data point with each other.

A perturbation algorithm $A(\cdot)$ probabilistically modifies the original data to other values in the same domain. It achieves ϵ-differential privacy, if and only if for any two adjacent datasets D, D' and any possible output τ of the perturbation algorithm, the following inequality always holds.

$$e^{-\epsilon} \leq \frac{Pr\left[A(D) = \tau\right]}{Pr\left[A(D') = \tau\right]} \leq e^{\epsilon}.$$

Intuitively, privacy budget ϵ controls how close the sanitized data is to the original one. A larger privacy budget will induce a higher degree of similarity as well as utility.

Duchi *et al.* [23] extended this notation to local differential privacy (LDP) for distributed data sources. Randomized response proposed by Warner *et al.* [92] is the baseline perturbation algorithm for LDP, which protects binary answers of individuals. Although differential privacy has not been used in model extraction and defense, it has been applied in several adversarial machine learning tasks. For example, Abadi *et al.* [2] introduced differentially private stochastic gradient descent to deep learning, which can preserve private information of the training set. Lee *et al.* [51] further improved its effectiveness using an adaptive privacy budget. Their approaches are shown effective against model inversion attack [26] or membership inference attack[81].

DP is defined in a centralized setting where data are collected and randomized by a trusted third-party, whereas LDP removes this role and assumes local randomization by users themselves before sending to the collector. As such, they are adopted for fundamentally different scenarios. We do not think that DP/LDP are naturally superior to other anonymity metrics. Instead, DP/LDP are developed from the perspective of theoretically provable bound of privacy by introducing randomization, which makes few assumptions on the type of attacks. Other anonymity metrics such as k-anonymity only have syntactic privacy notions. Although they provide privacy protection from different aspects, it has been indicated by recent study[56] that k-anonymity can satisfy DP with reasonable conditions.

2.4 Side-channel Inference and Localization

Side-channel attacks on mobile devices have evolved drastically. Since smart devices constantly sense the environmental information with their embedded sensors, external influence such as temperature, air pressure, noise, and body movement may create unique patterns on sensory data. For motion-related sensors, Owusu *et al.* [72] eavesdropped users' passwords using only accelerometer to detect the acceleration caused by different digits. Mehrnezhad *et al.* [60] further improved this approach by a website that can smuggle sensor readings with JavaScript. For radio frequency related sensors, Li *et al.* [55] proposed WindTalker to infer sensitive keystrokes on mobile devices with side-channel information from wireless network. As for other environmental sensing modules (e.g. barometer, magnetometer, microphone, and camera), similar approach has been applied to the inference of identity [101] and behavior [35].

Recently side-channel attacks have exploited the inertial sensors in Android and iOS to infer a user's outdoor location and even trace him/her. While we pay close attention to the indoor scenario with pedestrian where the estimation has to deal with limited data and volatile movement, most of related works focus on outdoor inference where multiple resources are available (e.g. GPS, street map and real-time public transport database). Users' driving routes have been successfully tracked in [33] and [66] by motion sensory information from their mobile devices. The former work is based on accelerometer and gyroscope, which leverages a dead reckoning technique with probability mapping algorithm while the latter one uses fine-grained gyroscope data and the graph of road information. Other than motion sensors, ambient sensors have also been investigated in driving route inference attack. Won *et al.* [95] proposed to use the latent relation between barometer readings and the geolocation to track drivers. As for public transportation, Hua *et al.* [36] showed that they could reveal users' daily metro schedule by monitoring accelerometers whose data are significantly affected by the route of metro. Watanabe *et al.* [93] demonstrated how to infer users' train schedule by matching user motions with the public railway database.

Other location attacks leverage non-sensory or active information. Mosenia *et al.* [64] can track users on train, plane or outdoor walking using hybrid sources. Michalevsky *et al.* [63] designed a location inference attack by profiling power consumption during commutes as cellular signal strength varies. Gao *et al.* [27] showed that usage-based automotive insurance can expose a driver's route through the recorded driving speed while Zhou *et al.* [100] enhanced the inference performance using real-time traffic and proposed defense framework with privacy-preserving scoring and audition. Kenneth *et al.* [7] develop an active location attack using signal transmitted from low power magnetic coil and received by mobile magnetometer. Cellular network based localization has also shown to be effective by observing the pattern of data transmission [83] or listening to GSM broadcast channel [49]. Li *et al.* [54] and Ometov *et al*. [70] proposed new methods on location tracking through social network footprints. Arp *et al*. [6] used the ultrasonic wave to infer a user's current location.

In traditional localization, mobile devices have played a major role in indoor positioning system (IPS) over the past decade. Significant approaches in IPS include dead reckoning, magnetic field fingerprinting, wireless multilateration and fingerprinting. Among these approaches, dead reckoning extensively utilizes inertial motion sensors for displacement and activity analysis. It predicts user's indoor route from an initial position with continuously measured speed and direction. Murata *et al.* [65] improve the performance of basic indoor dead reckoning with human activity knowledge such as age and environment when estimating step length. While Kang *et al.* [44] develop a fine-grained system to derive accurate walking parameters from inertial sensor data. But in dead reckoning, small error may easily accumulate and lead to erroneous results [11]. An alternative indoor positioning technique using inertial sensor is to leverage magnetic filed information. Magnetic field fingerprinting requires an offline mapping of magnetic intensity [30]. Wireless multilateration is another more stable positioning solution which derives time of arrival or direction of arrival information from external reference devices to pinpoint current location [11] while wireless fingerprinting captures the distribution of wireless signal strength [47].

State-of-the-art IPS usually incorporates various radio frequency signals (WLAN, RFID, Bluetooth, etc.) with context information from the floor plan, ambient sound/light sensing, magnetic field map to provide reliable positioning [19]. Under this legitimate scenario, unconstrained permission is granted to access privileged radio sensors, pedestrian initial state, computation power and floor plan data. However, such resources are unavailable for a stealthy attacker, who also has little domain knowledge of mobility analysis or magnetic map construction.

2.5 Distributed Data Analytics

As mentioned above, LDP has been widely applied in distributed data collection, such as crowdsourcing scenario. It has found main application in statistical analysis tasks, such as frequency estimation over categorical data. Erlingsson *et al.* proposed RAPPOR [25] for this task, which transforms a sensitive string into a Bloom filter and then applies the randomized response method [91] to perturb it. Marginal release has been studied in [99] under LDP, which is a potential alternative to produce synthetic data for machine learning task. Learning models using distributed resources have been proposed for distributed GPU settings [17]. While they focus on a highly-controlled network inside a data center, Google proposes federated machine learning for a loose federation of multiple mobile clients with scalable design [59] and develops secure aggregation using encryption scheme such as multi-party computation [8]. As for the system aspects, the architecture of federated learning discussed in this thesis is horizontal design [97], which enables easy unification with LDP on communication level [75]. Particularly, horizontal federated systems in mobile edge computing have started to study differentially private version by injecting noises to either SGD process or the final updates [57].

3 Defending Private Decision Boundary Against Extraction Attack

In this chapter, we begin our study on the attack surface of model and defend it against extraction attack on model prediction. The pervasive application of artificial intelligent has encouraged the boosting business of machine learning services, such as Microsoft Azure Face API, Google Cloud Speech-to-Text, and Amazon Comprehend. To train these high-quality machine learning models, service providers need to spend intense human labor and computation resources to acquire a large well-labeled datasets and tune training process. However, a prediction API call, which consists of a query and its response, can be vulnerable to adversarial attacks that disclose the internal states of these models. Particularly, a *model extraction* attack [86] is able to restore important model parameters using the rich information (e.g., model type, prediction confidence) provided by the prediction API. Once the model is extracted, an adversary can further apply model inversion attack [26] to learn the proprietary training data, compromising the privacy of data contributors. Another follow-up attack on the extracted model is *evasion attack* [74], [96], which avoids a certain prediction result by modifying its query. For example, a hacker modifies the executable binaries of a malware or the contents of a phishing email in order not to be detected by an antivirus or spam email filter.

Countermeasures against *model extraction* attacks have received increased attention but are still inadequate. One of them is to restrict rich information in the prediction API, for example, by rounding the prediction confidence value to a low granularity. However, even if the service provider completely eliminates this value in the prediction API, that is, to offer prediction label only, an adversary can still defeat this protection by issuing large number of fine-tuned queries and train a replica of the original model with great similarity [58], [74], [86]. The other countermeasure is to detect malicious extraction by monitoring feature coverage [46] or query distribution [41], and stop the service when a certain threshold is reached. However, since we cannot preclude user collusion, all queries and responses must be considered aggregately, which leads to significant false positive cases and eventually the early termination of service.

To address the disadvantages, in this chapter we propose a new countermeasure that obfuscates the output label of a prediction response. There are three main concerns when designing this obfuscation mechanism. First, the accuracy of prediction API is highly correlated with the degree of obfuscation — if obfuscation needs to be applied to most queries, the utility of the machine learning service will degrade severely. Second, the obfuscation mechanism should be independent of the underlying machine learning models and can tackle a category of boundary-probing attacks. Third, the obfuscation mechanism should be customizable. That is, it should allow user-defined parameters that can trade utility for model privacy or vice versa.

Our key observation is that many model extraction attacks exploit fine-tuned queries near the decision boundary of a machine learning model achieve optimal extraction performance[71], [73]. We treat decision boundary probing as an abstract and necessary condition of high-quality extraction attacks. The responses of these queries disclose the details of model parameters and therefore should be obfuscated with priority. To this end, we propose a **b**oundary **d**ifferentially **p**rivate **l**ayer (BDPL) for machine learning services. BDPL provides a parameterized approach to obfuscate responses whose queries fall in a predefined boundary-sensitive zone. The notion of differential privacy guarantees the responses of all queries in the boundary-sensitive zone are indistinguishable from one another. As such, adversary cannot learn the decision boundary no matter how many queries are issued to the prediction API. On the other hand, the majority perturbation falls within boundary-sensitive zone and out-of-zone query is less affected from obfuscation. In this way, we can make the best use of the obfuscation and retain high utility of the machine learning service. To summarize, our contributions in this chapter are as follows.

- We propose a new protection mechanism, namely, boundary differential privacy, against model extraction with fine-tuned queries while balancing service utility and model protection level.
- We develop an efficient method to identify queries in the boundary-sensitive zone, and design a perturbation algorithm called boundary randomized response for binary model to guarantee boundary differential privacy.
- We generalize binary defense to multiclass model and develop corresponding perturbation algorithm in a pairwise manner.

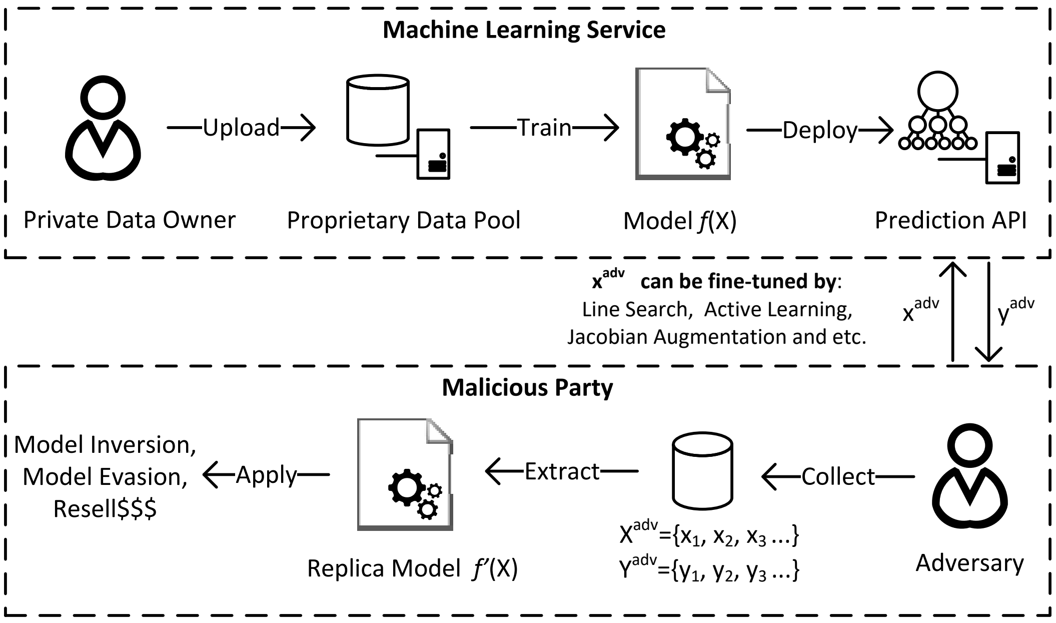


Fig. 3.1: Motivation and Threat Model

- We design an alternative defense layer with soft margin to extend the scope of protection and implement an adaptive perturbation algorithm.
- We conduct extensive empirical study on both binary and multiclass, linear and non-linear machine learning models to evaluate the effectiveness of our solution.

The rest of the chapter is organized as follows. Chapter 3.1 elaborates on the threat model and problem definition with boundary-sensitive zone and boundary differential privacy. Chapter 3.2 presents the details of boundary differentially private layer. Chapter 3.3 introduces evaluation metrics and shows the experimental results of BDPL against model extractions. Chapter 3.5 concludes this chapter and discusses future work.

3.1 Problem Definition

3.1.1 Motivation and Threat Model

A machine learning service provides a prediction result using a proprietary model as shown in Fig. 3.1. An adversary wants to produce a replica of this model by continuously querying it through the provided prediction API. We assume he can perform a typical two-stage extraction attack: 1) The adversary generates a set of fine-tuned real/synthetic queries normalized in [-1,1] and interacts with API under a large query budget. 2) He can store all responses, i.e., labels and reconstruct a replica model by training

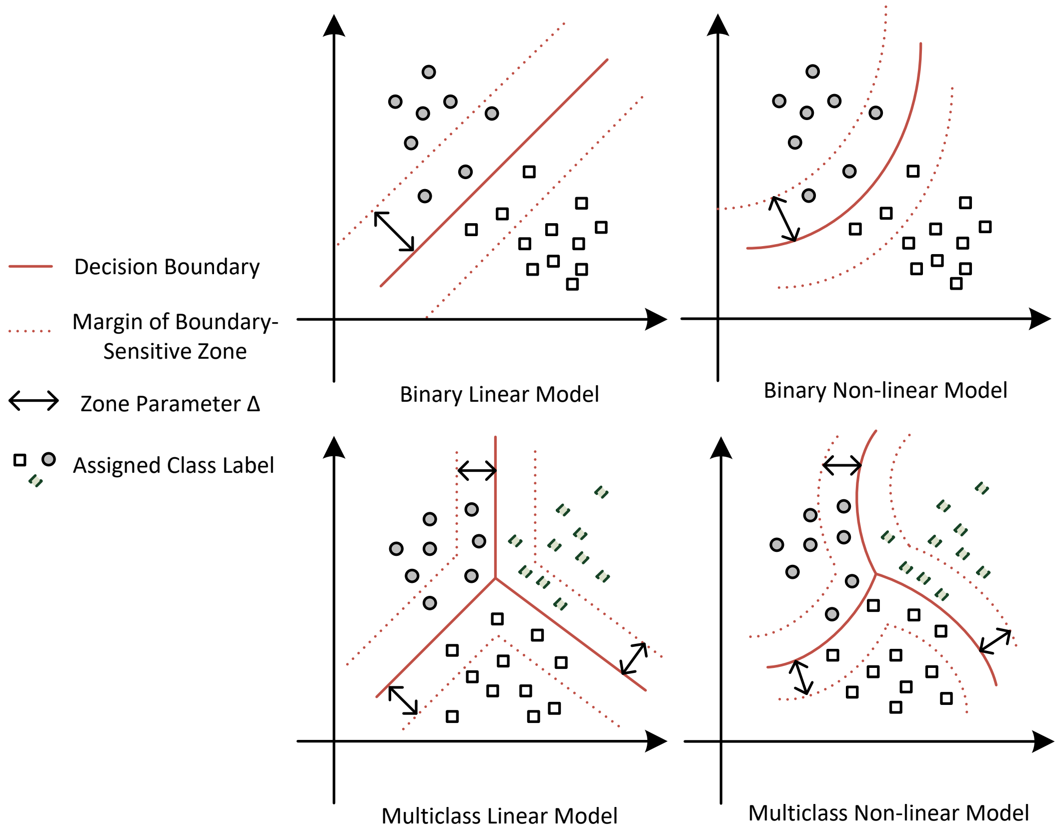


Fig. 3.2: Illustration of Hypothetical Decision Boundary and Boundary-Sensitive Zone in 2D

on the fine-tuned query-response pairs (e.g., minimize the empirical risk of an objective function). The success replication will results in intellectual property loss for the original provider and induce other attacks. The attack is semi-whitebox[1], i.e., it can extract a replicated model using the same model type (e.g., convolutional neural network) and hyperparameters as the original one. Apart from public knowledge of model and defense settings, we assume the adversary has no apriori information of the decision boundary.

3.1.2 Boundary-Sensitive Zone

Our problem is to protect against model extraction attacks by obfuscating query responses. Before we formally define the security model, we first introduce the notion of *decision boundary* and *boundary-sensitive zone*. For most supervised models, a decision boundary is a critical borderline in the feature space where labels are different on both sides. Fig. 3.2 illustrates the hypothetical decision boundaries in four combination cases in a 2D feature

[1]The semi-whitebox assumption is based on the fact that state-of-the-art models in specific application domains, such as image classification, are usually public knowledge. Nonetheless, our solution can also work against black-box attacks where such knowledge is proprietary.

space. In a multi-dimensional feature space, a line boundary becomes a hyperplane, and a curve boundary becomes a hypersurface.

Our key idea is to protect the query responses near the decision boundary against most model extraction attacks. To this end, we introduce the notion of boundary-sensitive zone.

Definition 1. *(Boundary-Sensitive Zone) Given feature space Z, a model f and a parameter Δ chosen by the model owner, all feature vectors adjacent to the decision boundary of f constitute a subspace Z_Δ of Z, where*

$$Z_\Delta = \{\boldsymbol{x} \in \mathbb{R}^d \mid dist(\boldsymbol{x}, f) < \Delta\},$$

where $dist(\cdot)$ measures the distance between a feature vector $\boldsymbol{x}$ and the decision boundary of f. All queries in this zone Z_Δ are considered particularly sensitive and have high risk of revealing the decision boundary of this model.

3.1.3 Boundary Differential Privacy

All queries in the boundary-sensitive zone need obfuscation, whose objective is to perturb the responses of any two sensitive queries so that they are indistinguishable for the adversary to determine the true decision boundary within this zone. To this end, we adopt the notion of differential privacy and formally define *boundary differential privacy* as follows.

Definition 2. *(ϵ-Boundary Differential Privacy) A perturbation algorithm $A(\cdot)$ achieves ϵ-boundary differential privacy, if and only if for any two queries $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ in the boundary-sensitive zone Z_Δ, the following inequality always holds for the true responses y_1 and y_2 and the perturbed ones $A(y_1)$ and $A(y_2)$.*

$$e^{-\epsilon} \leq \frac{Pr\left[y_1 = y_2 \middle| A(y_1), A(y_2)\right]}{Pr\left[y_1 \neq y_2 \middle| A(y_1), A(y_2)\right]} \leq e^{\epsilon}.$$

The above inequality guarantees that an adversary cannot deduce whether two perturbed responses $A(y_1)$ and $A(y_2)$ originate from the same $(y_1 = y_2)$ or different labels $(y_1 \neq y_2)$ with high confidence (controlled by ϵ). As such, the adversary cannot use fine-tuned queries, no matter how many they are, to find the decision boundary within the granule of boundary-sensitive zone.

3.2 Boundary Differentially Private Layer

In this section, we present our solution to protect against model extraction attacks with respect to ϵ-boundary differential privacy (ϵ-BDP) by appending a BDP layer to the model output. According to Definition 2, this layer consists of two major steps — identifying query sensitivity, and perturbing the responses of sensitive queries to satisfy BDP. In what follows, we first introduce the implement of binary defense with a technique to identify sensitive queries with the notion of *corner points* and a perturbation algorithm called *boundary randomized response* to guarantee ϵ-BDP. Then we generalize it to multiclass model with a technique to identify *counter class* for developing its perturbation algorithm *multiclass boundary randomized response*. Finally, we introduce a zone-less variant with soft margin to globalize the defense while retaining majority of obfuscation inside boundary sensitive zone, where the perturbation is provided by *adaptive boundary randomized response*.

3.2.1 Binary Defense

We start from binary models which have only two labels — positive and negative, which are particularly popular in spam filtering, malware detection, and disease diagnosis.

3.2.1.1 Identifying Sensitive Queries In Binary Model

A query is identified as sensitive if it falls in the boundary-sensitive zone according to Definition 1. However, in practice the decision boundary may not have a closed form (especially for complex models such as neural networks). In this subsection, we propose a method to determine if a query $\boldsymbol{x}_q$ is sensitive without deriving the boundary-sensitive zone. The idea is to test if a ball centered at $\boldsymbol{x}_q$ with radius Δ intersects with the decision boundary[2]. In theory, this is equivalent to finding if there exists a flipping point $\boldsymbol{x}'$ in the ball that has a different label from that of the query point $\boldsymbol{x}_q$. Formally,

Definition 3. *(Query Sensitivity) A query $\boldsymbol{x}_q$ is sensitive, if and only if:*

$$\exists \boldsymbol{x}' \in B(\boldsymbol{x}_q, \Delta), s.t., f(\boldsymbol{x}') \neq f(\boldsymbol{x}_q),$$

where $B(\boldsymbol{x}_q, \Delta) = \{\boldsymbol{x} \in \mathbb{R}^d \,| dist(\boldsymbol{x}, \boldsymbol{x}_q) \leq \Delta\}$ is the ball centered at $\boldsymbol{x}_q$ with radius Δ.

[2]The case of tangency is rarely reached in real life given that the feature space is usually continuous. For simplicity, we mainly consider intersection.

The above definition needs to test infinite number of points in the ball, which is infeasible. Nonetheless, we observe that if the ball is convex and small enough,[3] a sufficient condition of query $\boldsymbol{x}_q$ being sensitive is that at least one of the *corner points* in each dimension of this ball $B(\boldsymbol{x}_q, \Delta)$ is a flipping point. As such, the sensitivity of query $\boldsymbol{x}_q$ can be approximated by testing the labels of $2d$ corner points of $\boldsymbol{x}_q$ without false negatives. Furthermore, if the distance metric is the $L1$ distance (i.e., Manhattan distance), this is also a necessary condition, which means that testing corner points leads to the exact sensitivity. For example, given a two-dimensional query [a,b] and $L1$ radius Δ, if corner points [$a \pm \Delta$, b] and [a, $b \pm \Delta$] have flipping events, query [a,b] is sensitive. The following theorem proves this.

Theorem 1. *(Flipping Corner Theorem) A sufficient condition of query $\boldsymbol{x}_q$ being sensitive is that,*

$$\exists \;\; \boldsymbol{\Delta}_i \in \Delta \cdot \boldsymbol{I}, \;\; f(\boldsymbol{x}_q \pm \boldsymbol{\Delta}_i) \neq f(\boldsymbol{x}_q),$$

where $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{\Delta}_i$ is the projected interval on some dimension i, and $\boldsymbol{x}_q \pm \boldsymbol{\Delta}_i$ denotes the two corner points in dimension i. If the distance metric is the $L1$ distance, this equation is also a necessary condition.

Proof. Let $\boldsymbol{x}_i$ be one of the corner points in dimension i.

- (*Sufficient Condition*) For any $\boldsymbol{x}_i$, the decision boundary must exist between $\boldsymbol{x}_i$ and $\boldsymbol{x}_q$ where $f(\boldsymbol{x}_i) \neq f(\boldsymbol{x}_q)$. It intersects line $\boldsymbol{x}_i\boldsymbol{x}_q$ at point $\boldsymbol{b}_i$. As $\boldsymbol{x}_i$, $\boldsymbol{x}_q$ and $\boldsymbol{b}_i$ are on the same straight line, we have

$$dist(\boldsymbol{x}_i, \boldsymbol{b}_i) + dist(\boldsymbol{x}_q, \boldsymbol{b}_i) = dist(\boldsymbol{x}_i, \boldsymbol{x}_q) = \Delta.$$

  Since dist($\boldsymbol{x}_q$,f) is the minimum distance between $\boldsymbol{x}_q$ and any point on the decision boundary, we have

$$dist(\boldsymbol{x}_q, f) \leq dist(\boldsymbol{x}_q, \boldsymbol{b}_i) = \Delta - dist(\boldsymbol{x}_i, \boldsymbol{b}_i) < \Delta.$$

  According to Definition 1, query $\boldsymbol{x}_q$ is sensitive and this proves the sufficient condition.

- (*Necessary Condition for $L1$ Distance*) If $\boldsymbol{x}_q$ is a sensitive query, an $L1$-ball centered at $\boldsymbol{x}_q$ with radius Δ will be given by

$$B(\boldsymbol{x}_q, \Delta) = \{\boldsymbol{x} \in \mathbb{R}^d \;\middle|\; dist_{L1}(\boldsymbol{x}, \boldsymbol{x}_q) \leq \Delta\}. \tag{3.1}$$

[3]If Δ is small, the decision boundary near the ball can be treated as a hyperplane.

Let $\boldsymbol{b}_m$ be the point which is the closest to $\boldsymbol{x}_q$ on the decision boundary of f. According to Definition 3, we have

$$dist_{L1}(\boldsymbol{x}_q, \boldsymbol{b}_m) = dist_{L1}(\boldsymbol{x}_q, f) < \Delta.$$

Since $\boldsymbol{x}_q$ is sensitive, $\boldsymbol{b}_m$ must be inside this $L1$-ball:

$$\boldsymbol{b}_m \in B(\boldsymbol{x}_q, \Delta).$$

This means that the decision boundary must intersect the ball at $\boldsymbol{b}_m$. As such, at least one convex vertex of the ball is on a different side of the decision boundary than point $\boldsymbol{x}_q$. Since the convex vertices of an $L1$-ball are exactly those corner points, there exists at least one corner point $\boldsymbol{x}_i$ such that $f(\boldsymbol{x}_i) \neq f(\boldsymbol{x}_q)$. And this proves the necessary condition.

Therefore, flipping corner point is a sufficient condition for query $\boldsymbol{x}_q$ being sensitive and a necessary condition under the $L1$ distance metric. □

3.2.1.2 Perturbation Algorithm: Boundary Randomized Response

Randomized response [92] is a privacy-preserving survey technique developed for surveying sensitive questions. A randomized boolean value is given to the answer and provides plausible deniability. As the perturbation algorithm defined in boundary differential privacy has exactly two output choices, we design the following BRR algorithm based on randomized response to satisfy ϵ-BDP.

Definition 4. *(Boundary Randomized Response, BRR) Given query sample $\boldsymbol{x}_q$ and its true response $y_q \in \{0, 1\}$, the boundary randomized response algorithm $A(y_q)$ perturbs y_q by the following:*

$$A(y_q) = \begin{cases} y_q, & w.p. \quad \frac{1}{2} + \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}} \\ \\ 1 - y_q, & w.p. \quad \frac{1}{2} - \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}} \end{cases}.$$

Theorem 2. *The boundary randomized response algorithm $A(y_q)$ satisfies ϵ-BDP.*

Proof. To satisfy ϵ-BDP, the following inequality must hold according to Definition 2.

$$\frac{Pr[y_1 = y_2 | A(y_1), A(y_2)]}{Pr[y_1 \neq y_2 | A(y_1), A(y_2)]} \leq e^{\epsilon}. \tag{3.2}$$

We assume p is the probability of retaining y_q and $1-p$ the probability of flipping y_q. According to algorithm A, for any two responses $y_1, y_2 \in \{0,1\}$, the four possible cases for the above inequality are:

$$
\begin{aligned}
&\frac{Pr[y_1 = y_2 | A(y_1)=0, A(y_2)=0]}{Pr[y_1 \neq y_2 | A(y_1)=0, A(y_2)=0]}, or\\
&\frac{Pr[y_1 = y_2 | A(y_1)=1, A(y_2)=1]}{Pr[y_1 \neq y_2 | A(y_1)=1, A(y_2)=1]} = \frac{p^2+(1-p)^2}{2p(1-p)},\\
and\\
&\frac{Pr[y_1 = y_2 | A(y_1)=0, A(y_2)=1]}{Pr[y_1 \neq y_2 | A(y_1)=0, A(y_2)=1]}, or\\
&\frac{Pr[y_1 = y_2 | A(y_1)=1, A(y_2)=0]}{Pr[y_1 \neq y_2 | A(y_1)=1, A(y_2)=0]} = \frac{2p(1-p)}{p^2+(1-p)^2}.
\end{aligned}
$$

Given $0 \le p \le 1$, it is easy to prove that the former two cases are always larger than the latter. If we further use equality instead of inequality in Eqn. 3.2, we can derive the following equation of p:

$$\frac{p^2+(1-p)^2}{2p\cdot(1-p)} = e^{\epsilon}.$$

By solving the above equation, we can derive p as

$$
\begin{aligned}
p &= \frac{(2+2e^{\epsilon}) \pm \sqrt{(2+2e^{\epsilon})^2 - 4(2+2e^{\epsilon})}}{2(2+2e^{\epsilon})}.\\
p_1 &= \frac{1}{2} + \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}}, \; p_2 = \frac{1}{2} - \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}}. \qquad (3.3)
\end{aligned}
$$

Finally, we need to test the validity of both solutions. Let $u = e^{\epsilon}$, the derivative of p_1 in Eqn. 3.3 with respect to u is:

$$\frac{\partial p}{\partial u} = \frac{(\frac{2}{u-1})(\sqrt{u^2-1})}{(2+2u)^2} \ge 0.$$

As such, p_1 is monotonic with respect to u and ϵ. Since $\epsilon \in [0, +\infty]$, the lower and upper bounds of p_1 are obtained when $\epsilon = 0$ and $\epsilon = +\infty$:

$$
\begin{aligned}
&\lim_{\epsilon \to 0} \left[\frac{1}{2} + \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}}\right] = \frac{1}{2},\\
&\lim_{\epsilon \to +\infty} \left[\frac{1}{2} + \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}}\right] = \lim_{\epsilon \to +\infty} \left[\frac{1}{2} + \frac{\sqrt{1-\frac{1}{e^{2\epsilon}}}}{\frac{2}{e^{\epsilon}}+2}\right] = 1.
\end{aligned}
$$

Algorithm 1 Boundary Differentially Private Layer For Binary Model

```
Input:      Boundary-Sensitive Zone Parameter $\Delta$
            Boundary Privacy Budget $\epsilon$
            Query $\boldsymbol{x}_q \in R^d$
            Model $f$
Output:     Original Response $y_q$ or Perturbed Response $y'_q$
Procedure:
 1: if $\boldsymbol{x}_q$ is not cached then
 2:     $y_q = f(\boldsymbol{x}_q)$
 3:     $CornerPoints = getCornerPoints(\Delta, \boldsymbol{x}_q)$
 4:     for $\boldsymbol{x}_i$ in $CornerPoints$ do
 5:         if $\boldsymbol{x}_i$ is a flipping point then
 6:             $y'_q = BRR(y_q, \epsilon)$
 7:             $Cache(\boldsymbol{x}_q, y'_q)$
 8:             return $y'_q$
 9:     return $y_q$
10: else
11:     return $getCached(\boldsymbol{x}_q)$
```

As such, the derived p_1 in Eq. 3.3 is in the range of $[\frac{1}{2}, 1)$ and is thus valid. Similarly, we can prove p_2 is in the range of $(0, \frac{1}{2}]$ and is thus invalid. $\square$

3.2.1.3 Summary for Binary Defense

Algorithm 1 summarizes the detailed procedures of BDP layer that can be tapped to the output of any binary machine learning model f. When a new query $\boldsymbol{x}_q$ arrives, if it has already been queried before, the layer directly returns the cached response y'_q to prevent attacker from learning multiple perturbed responses of the same query response, which can lead to a less private BDP. Otherwise, the layer first obtains the real result y_q from model f. Then it determines whether $\boldsymbol{x}_q$ is in the boundary-sensitive zone by checking all corner points. As long as one corner point is as a flipping point, the query is identified as sensitive, and the boundary randomized response algorithm $BRR(\cdot)$ with privacy budget ϵ will be invoked. The layer will thus return the perturbed result y'_q and cache it for future use. Otherwise, if $\boldsymbol{x}_q$ is not sensitive after checking all corner points, the real result y_q will be returned. As for time complexity for identifying sensitive queries if they are not cached, since we only need to check two corner points in each dimension for an m-dimensional query, the upper bound time complexity will be $O(mT)$ where T is the time cost of each model prediction. Nonetheless, the average time cost can be much smaller as the process can terminate early as long as one flipping corner is found and we also propose n-shot sampling to speed up this process. As for T, in our evaluation environment we find the average of

T over a variety of models is $70 - 90\mu s$ for logistic regression, $400 - 500\mu s$ for shallow neural network, and $600 - 700\mu s$ for convolutional neural network.

3.2.2 Generalization to Multiclass Model

We now consider the case where the prediction domain of a model has more than two classes. To adapt the current algorithm to multiclass model and retain ϵ-BDP guarantee, we observe that the decision boundary in a multiclass model is essentially a union of binary boundaries, each of which separates two classes. As such, we can extend the definition of sensitive query in a multiclass model in terms of its nearby decision boundary. Formally,

Definition 5. *(Multiclass Query Sensitivity) A query $\boldsymbol{x}_q$ is sensitive to the decision boundary of classes $u, v \in Domain(f)$, if and only if:*

$$\exists \boldsymbol{x}' \in B(\boldsymbol{x}_q, \Delta),\ s.t., f(\boldsymbol{x}') = u, f(\boldsymbol{x}_q) = v, u \neq v,$$

where u is called the counter class to the true response v and $Domain(f)$ contains all possible output classes.

In this way, one class (i.e., true response) can be treated as the “positive” label and the other as the “negative” one (i.e., the counter class). The multiclass case is thus reduced to the same problem of protecting binary decision boundaries except that there are boundaries for each pair of classes.

3.2.2.1 Identifying Counter Class

The key idea of avoiding multiple decision boundaries from a variety of candidate counter classes for u is to only associate sensitive query with its nearest decision boundary and identify the corresponding class on the other side as the counter class. To identify this class, we use the majority vote from all flipping corner points. However, a full scan of all these points is not practical particularly in a high dimensional dataset with hundreds or even thousands of corner points. To strike a balance between accuracy and efficiency, we perform an N-shot sampling over the flipping corners. Formally,

Definition 6. *(N-shot Flipping Corner) An estimate of query $\boldsymbol{x}_q$ being sensitive to the decision boundary of classes $u, v \in Domain(f)$ is that,*

$$\forall\ \boldsymbol{\Delta}_{i \in N} \in \Delta \cdot \boldsymbol{I},\ V(f(\boldsymbol{x}_q \pm \boldsymbol{\Delta}_i) \neq f(\boldsymbol{x}_q)) = u,$$

where $\mathbf{\Delta}_{i\in N}$ denotes flipping corner points in N sampling dimensions with flipping corners and $V(\cdot)$ finds the class with the highest flipping rate from the comparison results of provided corner points.

3.2.2.2 Multiclass Boundary Randomize Response

Now that we can detect a sensitive query in the multiclass case, given its true response and counter class, we use the following Multiclass Boundary Randomized Response (MBRR) algorithm to achieve ϵ-BDP.

Definition 7. *(Multiclass Boundary Randomized Response, MBRR) Given a query sample $\boldsymbol{x}_q$, its true response y_q and counter class $\overline{y_q}$ (y_q, $\overline{y_q} \in \{u, v\}$), the multiclass boundary randomized response algorithm $A(y_q)$ perturbs y_q by the following:*

$$A(y_q) = \begin{cases} y_q, & w.p. \quad \frac{1}{2} + \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}} \\ \\ \overline{y_q}, & w.p. \quad \frac{1}{2} - \frac{\sqrt{e^{2\epsilon}-1}}{2+2e^{\epsilon}} \end{cases}.$$

Theorem 3. *The multiclass boundary randomized response algorithm $A(y_q)$ provides ϵ-BDP to each binary decision boundary $f_{u,v}$ in the model.*

The proof is similar to that of Theorem 2 and is thus omitted.

3.2.2.3 Summary for Multiclass Defense

Algorithm 2 summarizes the detailed procedures of BDP layer for a multiclass machine learning model f. Similar to the binary model, the layer directly returns a cached response to retain privacy guarantee if query $\boldsymbol{x}_q$ has been processed before. In addition to zone parameter Δ and privacy budget ϵ, the number of samples N to determine the counter class can be tuned between efficiency and accuracy. After the counter class is determined, the multiclass boundary randomized response algorithm $MBRR(\cdot)$ with privacy budget ϵ is invoked. The layer then returns the perturbed result y'_q and caches it for future use.

3.2.3 Zone-less Boundary Differentially Private Layer

In the previous section, the decision boundary is formulated as a zone with a hard margin controlled by Δ — a query is either inside this zone (i.e.,

Algorithm 2 Boundary Differentially Private Layer For Multiclass Model

```
Input:      Flipping Corner Shot $N$
            Boundary-Sensitive Zone Parameter $\Delta$
            Boundary Privacy Budget $\epsilon$
            Query $\boldsymbol{x}_q \in R^d$
            Model $f$
Output:     Original Response $y_q$ or Perturbed Response $y'_q$
Procedure:
 1: if $\boldsymbol{x}_q$ is not cached then
 2:     $y_q = f(\boldsymbol{x}_q)$
 3:     $cPoints = getCornerPoints(\Delta, \boldsymbol{x}_q, N)$
 4:     for $\boldsymbol{x}_i$ in $CornerPoints$ do
 5:         if $\boldsymbol{x}_i$ is a flipping point then
 6:             $CounterClass = getCounterClass(cPoints)$
 7:             $y'_q = MBRR(y_q, \epsilon, CounterClass)$
 8:             $Cache(\boldsymbol{x}_q, y'_q)$
 9:             return $y'_q$
10:     return $y_q$
11: else
12:     return $getCached(\boldsymbol{x}_q)$
```

sensitive) or outside of it (i.e., non-sensitive). ϵ-BDP can be achieved in the former case but no privacy is provided in the latter case. This makes the choice of Δ a crucial and challenging task for the user — a small value leaves some decision boundary unprotected and yet a large value introduces unnecessary noise to non-boundary area where no protection is needed. To make Δ less influential, in this section we propose a soft margin approach as an alternative to the hard margin.

3.2.3.1 Soft Query Sensitivity

The soft margin is essentially defined through the notion of soft query sensitivity, in which a query is no longer a hard "0" (non-sensitive) or "1" (sensitive). Instead, it is 1 on the soft margin and is larger than 1 when inside the margin. Then the degree of perturbation depends on the query sensitivity. The rationale behind a soft sensitivity of a query is three-folded. First, it must have a negative correlation with its distance to the nearest decision boundary, because query results reveal more information about the boundary and thus are more sensitive when they are closer to it. Second, how much the sensitivity depends on the distance should be controlled by the model owner. Third, the soft and hard sensitivity should be a unified notion. That is, the perturbation protocol for the hard sensitivity, Boundary Randomized Response (BRR), must still work with minimum adaptation. The following is a definition that satisfies all three rationales.

Definition 8. *(Soft Query Sensitivity) Given a model f and a zone parameter Δ chosen by the model owner, the sensitivity of a query $\boldsymbol{x_q}$ is a fractional function as:*

$$s(\boldsymbol{x_q}) = \frac{\Delta}{dist(\boldsymbol{x_q}, f_{u,v})}, \tag{3.4}$$

where $dist(\boldsymbol{x_q}, f_{u,v})$ measures the distance between query $\boldsymbol{x_q}$ and the nearest decision boundary $f_{u,v}$.

To derive the distance without a closed form of the decision boundary, we can still adopt the flipping-corner-point method. The idea is to perform a binary search with an initial distance guess. If flipping corner points occur, the distance must be smaller, so we reduce the current guess to half and repeat the search; otherwise we double the guess. The final distance is obtained when a precision threshold or a maximum number of iterations is reached.

3.2.3.2 Adaptive Boundary Randomized Response

Given the above definition of query sensitivity, the perturbation algorithm, Adaptive Boundary Randomized Response (ABRR), is exactly the same as BRR, except for the exponents. In BRR, the exponent is ϵ which is implicitly $\frac{\epsilon}{s(\boldsymbol{x_q})}$ where $s(\boldsymbol{x_q})$ is always 1. ABRR uses the same formulae where $s(\boldsymbol{x_q})$ is defined in Eqn. 3.4.

Definition 9. *(Adaptive Boundary Randomized Response, ABRR) Given query sample $\boldsymbol{x}_q$ normalized to [-1,1], query sensitivity $s(\boldsymbol{x_q})$, its true response y_q and counter class $\overline{y_q}$ (y_q, $\overline{y_q} \in \{u, v\}$), the adaptive boundary randomized response algorithm $A(y_q)$ perturbs y_q by the following:*

$$A(y_q) = \begin{cases} y_q, & \text{w.p.} \quad \frac{1}{2} + \frac{\sqrt{e^{2\psi}-1}}{2+2e^{\psi}} \\ \\ \overline{y_q}, & \text{w.p.} \quad \frac{1}{2} - \frac{\sqrt{e^{2\psi}-1}}{2+2e^{\psi}} \end{cases},$$

where $\psi = \frac{\epsilon}{s(\boldsymbol{x_q})}$.

Theorem 4. *The adaptive boundary randomized response algorithm $A(y_q)$ satisfies ψ-BDP to each binary decision boundary $f_{u,v}$, where $\psi \leq \frac{2}{\Delta}\epsilon$.*

Proof. We assume p_1, p_2 are the probabilities of retaining true responses for two queries $\boldsymbol{x}_1, \boldsymbol{x}_2$. According to ABRR, for any two responses $y_1, y_2 \in \{u, v\}$, the four possible cases to derive the BDP inequality are:

$$\begin{aligned}
&\frac{Pr[y_1 = y_2 | A(y_1) = u, A(y_2) = u]}{Pr[y_1 \neq y_2 | A(y_1) = u, A(y_2) = u]}, or \\
&\frac{Pr[y_1 = y_2 | A(y_1) = v, A(y_2) = v]}{Pr[y_1 \neq y_2 | A(y_1) = v, A(y_2) = v]} = \frac{p_1 p_2 + (1-p_1)(1-p_2)}{p_1(1-p_2) + p_2(1-p_1)}, \\
and \\
&\frac{Pr[y_1 = y_2 | A(y_1) = u, A(y_2) = v]}{Pr[y_1 \neq y_2 | A(y_1) = u, A(y_2) = v]}, or \\
&\frac{Pr[y_1 = y_2 | A(y_1) = v, A(y_2) = u]}{Pr[y_1 \neq y_2 | A(y_1) = v, A(y_2) = u]} = \frac{p_1(1-p_2) + p_2(1-p_1)}{p_1 p_2 + (1-p_1)(1-p_2)}.
\end{aligned}$$

Since $p_1, p_2 \in [\frac{1}{2}, 1)$, the partial derivatives to p_1 and p_2 of the right-hand side term in the former two cases are

$$\frac{\partial \frac{p_1 p_2 + (1-p_1)(1-p_2)}{p_1(1-p_2)+p_2(1-p_1)}}{\partial p_1} = \frac{2p_1 - 1}{(-2p_1p_2 + p_1 + p_2)^2} \geq 0,$$

$$\frac{\partial \frac{p_1 p_2 + (1-p_1)(1-p_2)}{p_1(1-p_2)+p_2(1-p_1)}}{\partial p_2} = \frac{2p_2 - 1}{(-2p_1p_2 + p_1 + p_2)^2} \geq 0.$$

As such, the right-hand side term in the former two cases is monotonically increasing. Similarly, that term in the latter two cases is monotonically decreasing. Let $p_{max} = \max\{p_1, p_2\}$. Then the two terms are bounded as follows.

$$\frac{p_1 p_2 + (1-p_1)(1-p_2)}{p_1(1-p_2) + p_2(1-p_1)} \leq \frac{{p_{max}}^2 + (1-p_{max})^2}{2p_{max}(1-p_{max})}, \tag{3.5}$$

$$\frac{p_1(1-p_2) + p_2(1-p_1)}{p_1 p_2 + (1-p_1)(1-p_2)} \leq 1. \tag{3.6}$$

Furthermore, since $p_1, p_2 \in [\frac{1}{2}, 1)$, the right-hand side term of Eqn. 3.5 also serves as the upper bound of the right-hand side term of Eqn. 3.6. That is,

$$\frac{{p_{max}}^2 + (1-p_{max})^2}{2p_{max}(1-p_{max})} \geq 1.$$

Algorithm 3 Zone-less Boundary Differentially Private Layer for Multiclass Model

```
Input:      Query $x_q \in R^d$
            Model $f$
            Soft Margin $\Delta$
            Boundary Privacy Budget $\epsilon$
Output:     Perturbed Response $y'_q$
Procedure:
1: if $x_q$ is not cached then
2:     $y_q = f(x_q)$
3:     $s_q = getSensitivity(f, x_q, \Delta)$
4:     $CounterClass = getCounterClass(f, x_q)$
5:     $y'_q = ABRR(y_q, \epsilon, s_q, CounterClass)$
6:     $Cache(x_q, y'_q)$
7:     return $y'_q$
8: else
9:     return $getCached(x_q)$
```

According to ABRR, we can derive p_{max} as

$$p_{max} = \frac{\sqrt{e^{2\psi_{max}} - 1}}{2 + 2e^{\psi_{max}}}.$$

By replacing p_{max} in the right term of Eqn. 3.5 with it, we have

$$\frac{{p_{max}}^2 + (1 - p_{max})^2}{2p_{max}(1 - p_{max})} = e^{\psi_{max}}.$$

Finally, since the sensitivity $s(\boldsymbol{x_q})$ is in the range $[\frac{2}{\Delta}, +\infty]$, we derive the bound of ψ_{max} as

$$\psi_{max} \leq \frac{\epsilon}{s(\boldsymbol{x_q})} \leq \frac{\Delta}{2}\epsilon. \tag{3.7}$$

Due to the monotonicity of exponential function, we have

$$e^{\psi_{max}} \leq e^{\frac{2}{\Delta}\epsilon}. \tag{3.8}$$

Therefore, for any two queries, the algorithm satisfies ψ-BDP where $\psi \leq \frac{2}{\Delta}\epsilon$. $\square$

Notably, for queries inside the margin of Δ, sensitivity is equal or greater than 1. As a result, we can prove that a minimum of ϵ-BDP is always achieved, same as the requirement for boundary-sensitive zone in BDPL. In other words, ABRR essentially provides stronger non-uniform ϵ-BDP than BRR in Δ boundary-sensitive zone. We summarize this property as follows.

Corollary 1. *For any query $\boldsymbol{x}_q$ inside the margin of Δ, the adaptive boundary randomized response algorithm $A(y_q)$ satisfies ψ-BDP where $\psi \leq \epsilon$*

Proof. Since query $\boldsymbol{x}_q$ now has $s(\boldsymbol{x}_q)$ in the range $[1, +\infty]$, , we can prove the following by Eqn. 3.7 and 3.8.

$$e^{\psi_{max}} \leq e^{\epsilon}.$$

□

3.2.3.3 Summary for Zone-less Defense

Algorithm 3 summarizes the detailed procedures of zone-less BDP layer with soft margin. Caching policy is still carried out for any historical query. If $\boldsymbol{x}_q$ is a new query, the sensitivity of $\boldsymbol{x}_q$ to nearest decision boundary is first measured using a binary search with the corner-point technique. Then the counter class is calculated for a multiclass model. Finally, the adaptive boundary randomized response algorithm $ABRR(\cdot)$ is invoked to perform perturbation with BDP protection.

3.3 Experiments

In this section, we evaluate the effectiveness of boundary differentially private layer (BDPL) against model extraction attacks. Specifically, we implement those motivating extraction attacks using fine-tuned queries as in [58], [86] and compare the success rates of these attacks with and without BDPL.

3.3.1 Setup

3.3.1.1 Datasets and Machine Learning Models

We evaluate three datasets and two models used in the literature [86] — a Botany dataset *Mushrooms* (113 attributes, 8124 records), a census dataset *Adult* (109 attributes, 48842 records) and a general social survey dataset *GSS* (101 attributes, 16127 records). The former two datasets are obtained from UCI machine learning repository [22] while the last one is from NORC[82]. All categorical items are processed by *one-hot-encoding* [34] and missing values are replaced with the mean value of this attribute. We adopt *min-max normalization* to unify all feature domains into [-1,1]. Data augmentation is not used for all the experiments, in accordance with the configuration of the original attacks.

For the evaluation of binary defense, *Mushrooms* dataset is trained on the label that shows whether a mushroom is poisonous or edible, and *Adult* dataset is trained on the label that shows whether the annual income of an adult exceeds 50K (Adult-b). As for multiclass defense, *GSS* dataset is used to train a model to predict level of happiness while *Adult* dataset is used to predict the race of the participants (Adult-m).

We train both a linear model, namely, logistic regression (LR), and a non-linear model, namely, 3-layer neural network (NN), to predict unknown labels on the above datasets. Logistic regression is implemented using *cross-entropy* loss with $L2$ regularizer. Neural network is implemented using TensorFlow r1.12 [1]. The hidden layer contains 20 neurons with $tanh$ activation. The output layer is implemented with a $sigmoid$ function for binary prediction and a $softmax$ function for multiclass prediction.

3.3.1.2 Attack and Evaluation Metrics

We implement the extraction attack defined in Chapter **??** using original attack code of line-search technique in [86]. Specifically, the attacker first creates a seed set of pairwise queries with opposite or different response labels, and then searches for new samples that lie on the line segment connecting this pair to approach the decision boundary. This process is repeated until either a searching threshold or query limit is reached. The size of a seed set is 4 and the searching threshold is 0.05. It is a white-box attack which produces an extracted model f' with the same hyperparameters and architectures as the original model f. To compare f and f', we adopt *extraction rate* [46], [86] to measure the proportion of matching predictions (i.e., both f and f' predict the same label) in an evaluation query set. Formally,

- **Extraction Rate (R)**. Given an evaluation query set $\mathcal{X}_e$, the extraction rate

$$R = \frac{1}{|\mathcal{X}_e|} \sum_{\boldsymbol{x}_i \in \mathcal{X}_e} \mathbb{1}(f(\boldsymbol{x}_i) = f'(\boldsymbol{x}_i)),$$

  where $\mathbb{1}(\cdot)$ is an indicator function that outputs 1 if the input condition holds and 0 otherwise. The extraction rate essentially measures the similarity of model outputs given the same inputs.

- **Utility (U)**. This second metric is evaluated on the test data points and measures the proportion of responses that are perturbed (i.e., flipped) by BDPL. It indicates how useful these responses are from a normal user's perspective. Formally, given the entire queries $\mathcal{X}_q$ issued from test set by clients, and the set of (perturbed) responses $\mathcal{Y}_q$ from the service provider,

$$U = \frac{1}{|\mathcal{X}_q|} \sum_{\boldsymbol{x}_i \in \mathcal{X}_q, y_i \in \mathcal{Y}_q} \mathbb{1}(f(\boldsymbol{x}_i) = y_i).$$

We follow the same evaluation setting as the original works. In the classic attack (Tramer's [86]) of Section 5.2-5.5, training set sample are used for the construction of victim model, whereas test set samples and uniformly sampled points are applied in the evaluation of utility and extraction rate respectively. In the advanced attack (ActiveThief [73]) of Section 5.6, the configuration is similar, except that the extraction rate is evaluated against test set samples.

3.3.2 Overall Evaluation

To evaluate how well the decision boundary can be protected by our solution, we launch extraction attacks on a number of model/dataset combinations and plot the extraction rate R of sensitive queries in Figs. 3.3 and 3.4 in terms of the number of queries.

Evaluation of BDPL. In this experiment, we set $\Delta = 1/8$, and $\epsilon = 0.01$ for all models. As shown in Fig. 3.3, except for the initial extraction stage (query size less than $5K$), BDPL exhibits a significant protection effect for all 8 combinations — up to 12% drop on R — compared with no defense. The drops in *GSS w/ NN* and *Adult-m w/ NN* are smaller (around $5\%-6\%$) because these two models are the most complicated (multiclass neural networks) and the least vulnerable to label perturbation.

The secondary axis of Fig. 3.3 also plots the utility of BDPL using bar chart. We observe that the utility saturates at over 80% after $20K$ queries in all combinations (among which 4 can achieve nearly 90% utility) except for *Adult w/ LR*. This model has the fewest parameters and feature inputs, so BDPL has to perturb more sensitive queries to retain the same BDP level as the others. The impact on utility by Δ and ϵ will be further discussed in Section 3.3.4.

It is noteworthy that 1% reduction of extraction rate is more significant in later attack stage than earlier stage where the attackers need to increase the amounts of queries tremendously. For example, in Fig.6(e), the adversary spends $15K$ queries to improve extraction rate from 90% to 97%, which is canceled off by BDPL using after $15K$ queries with only 11% utility loss.

Obviously, 7% drop on extraction rate is more significant than an 11% utility loss because the former costs 15K queries whereas the latter costs only about 1.5K queries.

Evaluation of Zone-less BDPL. In this experiment, we set $\Delta = 1/8$ and increase ϵ to 0.16 so that the overall utility is similar to the previous experiment (i.e., over 80%). The experiment results on Mushrooms and GSS are plotted in Fig. 3.4. Zone-less BDPL provides even better protection in all 4 combinations than BDPL, particularly at the initial stage (query size less than $5K$) with a much lower extraction rate. Furthermore, all extraction rates saturate even earlier (after $10k$ of queries) than BDPL. Overall, we observe that zone-less BDPL performs particularly well with logistic regression models, where we witness an extra drop of 4% on R compared with BDPL. The impact on R and U by Δ and ϵ will be shown in Section 3.3.5.

3.3.3 BDPL vs. Uniform Perturbation

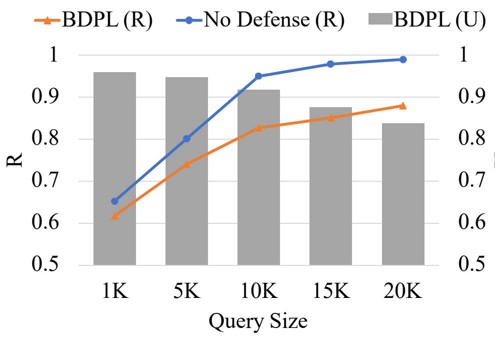
In this experiment, we compare BDPL on binary model (single decision boundary) with a uniform perturbation mechanism that randomly flips the response label by a certain probability, whether the query is sensitive or not. To have a fair comparison, we use trial-and-error[4] to find this probability so that the overall extraction rates of both mechanisms are almost the same. In Fig. 3.5, we plot the extraction rates of both mechanisms for *Mushrooms w/ LR* with $\Delta = 1/8$ and $\epsilon = 0.01$. We observe that BDPL outperforms uniform perturbation by 5%-7% extraction rate, which is very significant as this leads to an increase of misclassification rate by 30%-50%. As such, we can conclude that BDPL is very effective in protecting the decision boundary by differentiating sensitive queries from non-sensitive ones, and therefore it retains high utility for query samples that are faraway from the boundary.

3.3.4 Impact of ϵ and Δ in BDPL

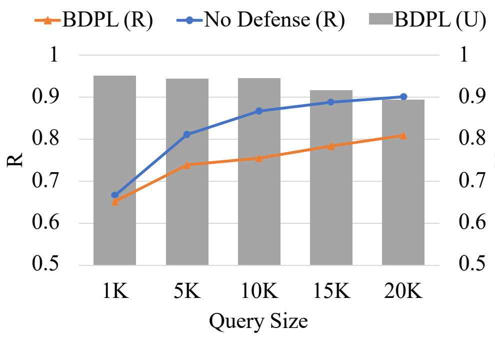
In this subsection, we evaluate BDPL performance with respect to zone parameter Δ and privacy budget ϵ. In Fig. 3.6, we fix ϵ and vary Δ from $1/64$ to $1/8$ for all 8 model/dataset combinations. In Fig. 3.7, we fix Δ and vary ϵ from 0.01 to 0.64 for all 8 model/dataset combinations.

Impact on Extraction Rate When Δ increases from $1/64$ to $1/8$, the extraction rate is significantly reduced in both logistic regression (up to

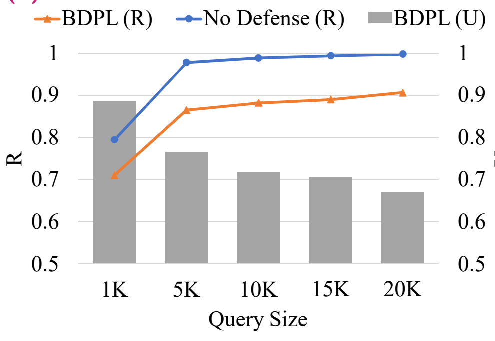
[4]To do this, we start with 1 random flip out of all responses and measure its overall extraction rate. We then repeatedly increment this number by 1 until the overall extraction rate is very close to that of BDPL.



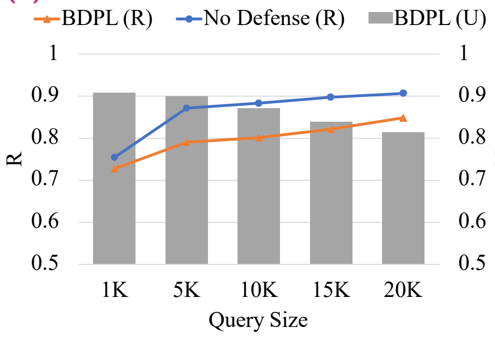
(a) Mushrooms w/ LR



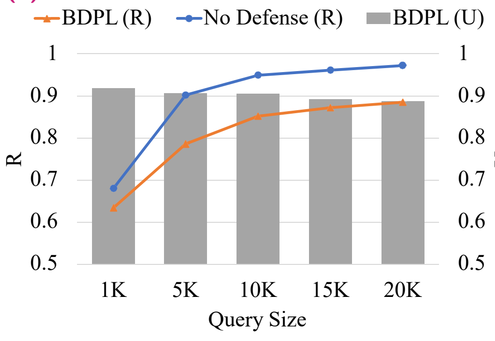
(b) Mushrooms w/ NN



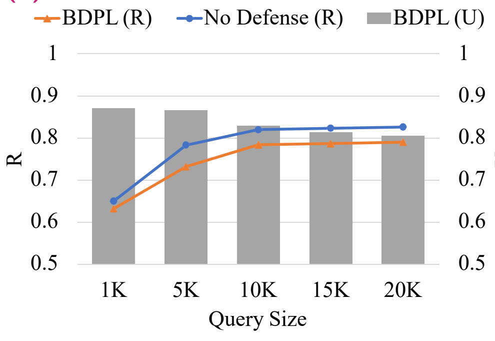
(c) Adult-b w/ LR



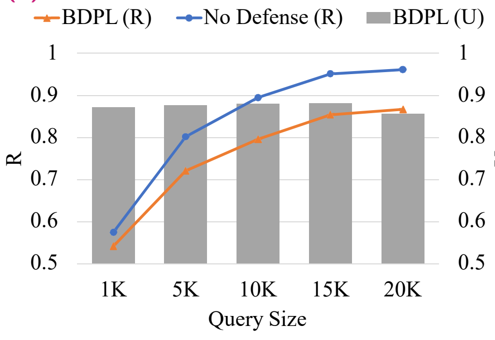
(d) Adult-b w/ NN



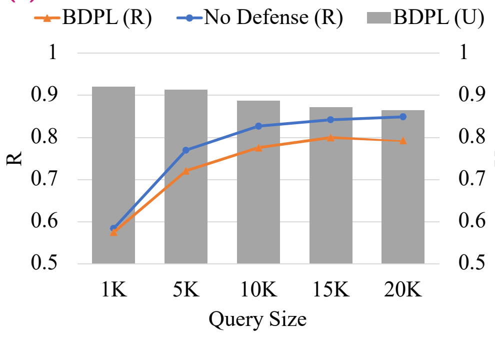
(e) GSS w/ LR



(f) GSS w/ NN

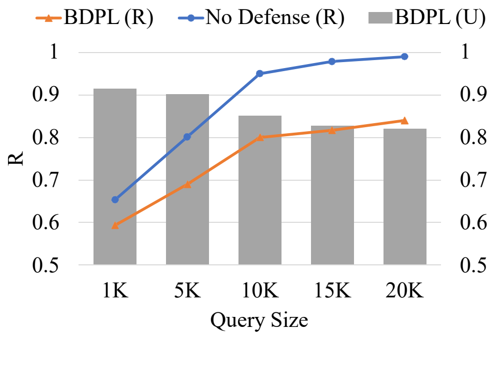


(g) Adult-m w/ LR

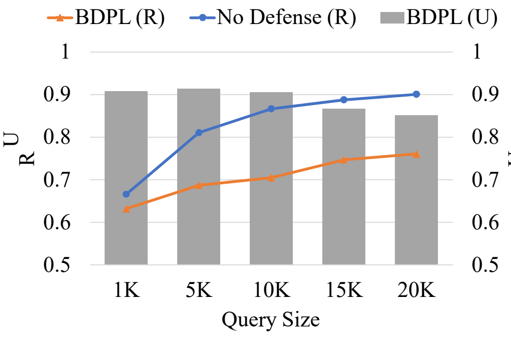


(h) Adult-m w/ NN

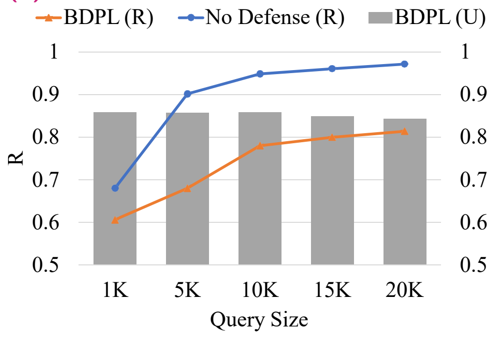
Fig. 3.3: Overall Protection Effect by BDPL: Extraction Rate and Utility



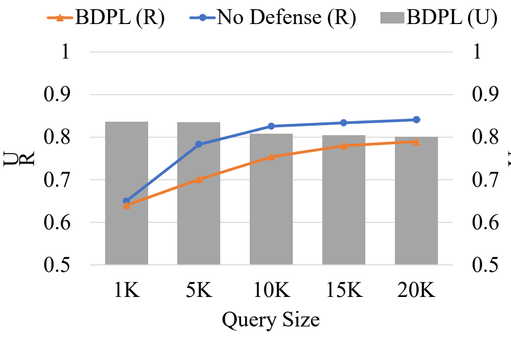
(a) Mushrooms w/ LR



(b) Mushrooms w/ NN



(c) GSS w/ LR



(d) GSS w/ NN

Fig. 3.4: Overall Protection Effect by Zone-less BDPL: Extraction Rate and Utility

12% drop) and neural network (up to 10% drop). Nonetheless, for neural networks, the extract rate does not change much when Δ increases from $1/64$ to $1/32$, which indicates that if the boundary-sensitive zone is too small, BDPL may not provide effective protection, especially when the decision boundary is non-linear. As for privacy budget ϵ, its impact is not as significant as Δ. We only observe up to 4% drop of extraction rate when ϵ decreases from 0.64 to 0.01 for all 8 model/dataset combinations.

Last but not the least, the extraction rates under all these settings saturate as the query size increases. In most cases, they start to saturate before $5K$

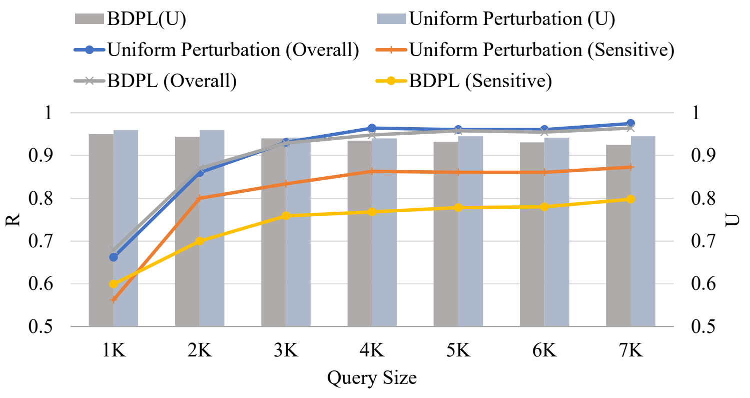


Fig. 3.5: BDPL vs. Uniform Perturbation

queries, and in the worst case, they saturate at $15K$ or $20K$. This indicates that BDPL imposes a theoretical upper bound on the extraction rate no matter how many queries are issued.

Impact on Utility In this part, we evaluate BDPL performance regarding utility under similar varying settings. In Fig. 3.8, we plot the final utility with respect to Δ and ϵ after $20K$ queries for all model/dataset combinations. Except for *Adult w/ LR*, all utilities are higher than 80% and most of them are above 90%, which means that BDPL does not severely sacrifice the accuracy of a machine learning service. As expected, the utility reaches peak when $\Delta = 1/64$ (the smallest zone size) and $\epsilon = 0.64$ (the least probability of perturbation). Furthermore, as is coincided with the extraction rate, the utility is more sensitive to Δ than to ϵ. For example, an increase of Δ from 0.01 to 0.1 leads to a drop of utility by 10%, whereas a decrease of ϵ from 0.1 to 0.01 leads to only 5% drop.

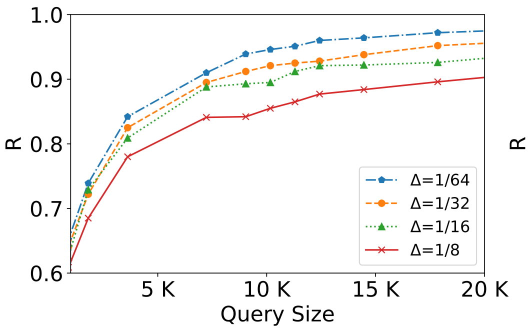
To conclude, BDPL permanently protects decision boundary of both linear and non-linear models with moderate utility loss. The changes of Δ and ϵ (particularly the former) have modest impact on the extraction rate and utility.

3.3.5 Impact of ϵ and Δ in Zone-less BDPL

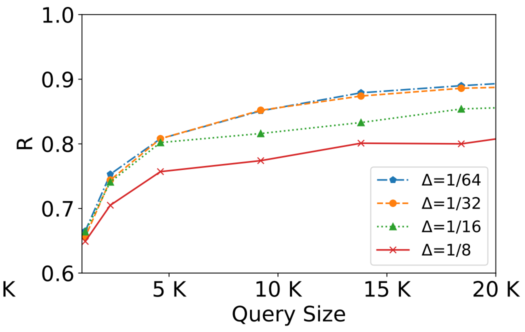
In this subsection, we evaluate zone-less BDPL performance with respect to Δ and ϵ. In Fig. 3.9, we fix ϵ and vary Δ from $1/64$ to $1/8$. In Fig. 3.10, we fix Δ and vary ϵ from 0.01 to 0.64. Due to space limitation, we only plot the results on both dataset/model combinations, i.e., *GSS w/ LR* and *GSS w/ NN*.

Impact on Extraction Rate. Both parameters maintain effectiveness in protecting decision boundary and saturating the extraction rate. Particularly, compared to the hard margin solution, when ϵ decreases from 0.64 to 0.01, zone-less BDPL draws significant drop over extraction rate (up to 25% drop in logistic regression and 15% in neural network). This coincides with Corollary 1 in Section 3.2.3 that zone-less BDPL achieves better ϵ-BDP protection than BDPL. Meanwhile, varying zone parameter Δ has less eminent effect than in the hard margin case. This coincides with our zone-less design to protect the decision boundary with a soft margin.

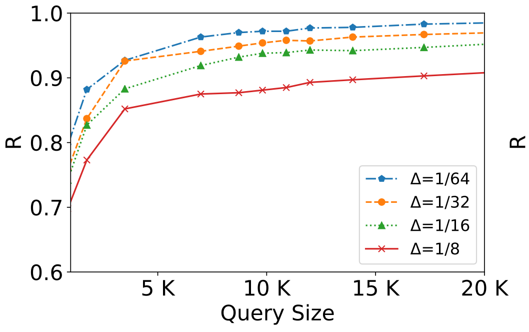
Impact on Utility. We evaluate zone-less BDPL in terms of utility after $20K$ queries. In Fig. 3.11, we observe that the change of ϵ leads to 35% change of utility while the change of Δ only leads to 10%. This can be explained



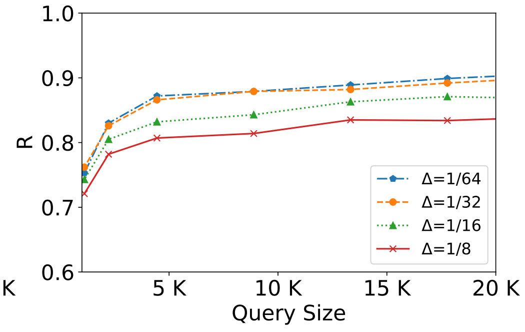
(a) Mushrooms w/ LR



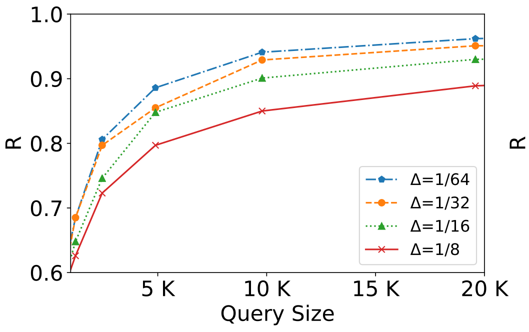
(b) Mushrooms w/ NN



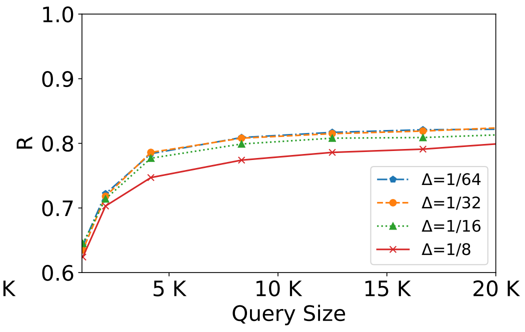
(c) Adult-b w/ LR



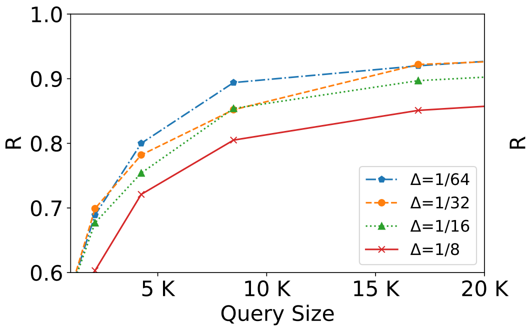
(d) Adult-b w/ NN



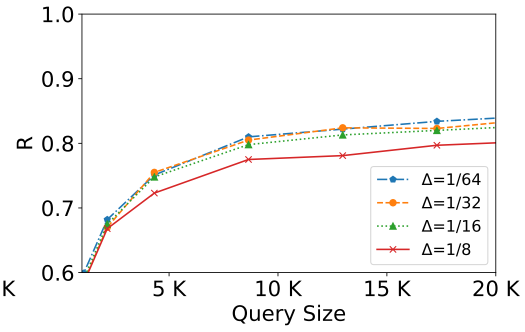
(e) GSS w/ LR



(f) GSS w/ NN

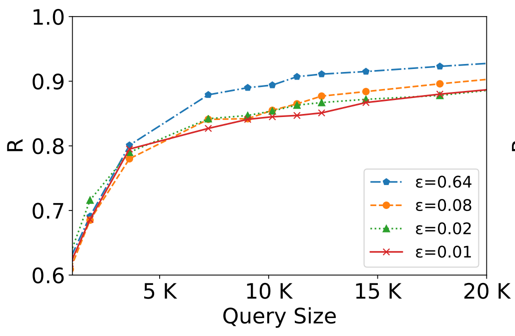


(g) Adult-m w/ LR

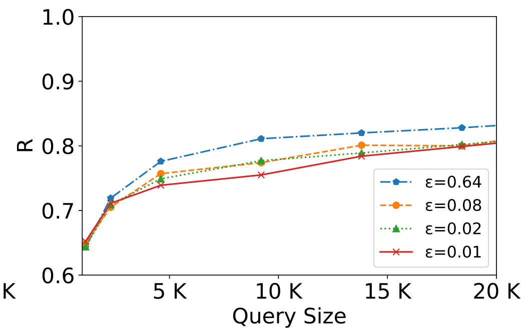


(h) Adult-m w/ NN

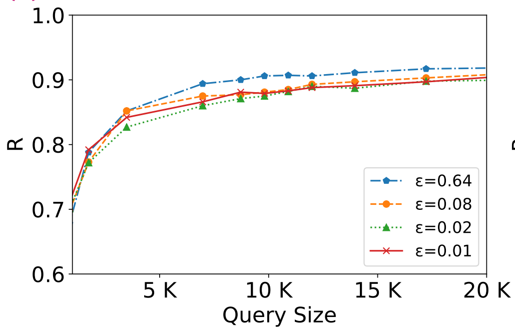
Fig. 3.6: Impact of Varying Δ in BDPL with $\epsilon = 0.01$



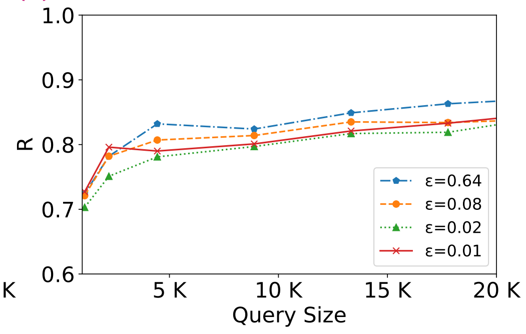
(a) Mushrooms w/ LR



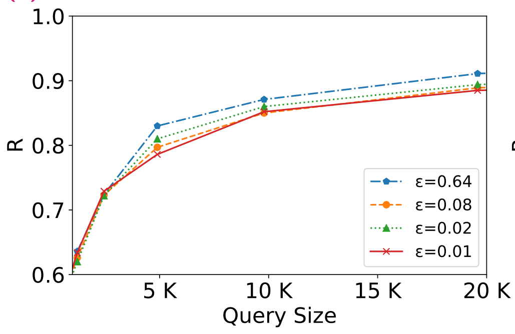
(b) Mushrooms w/ NN



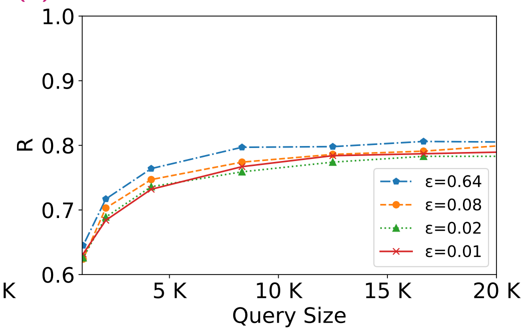
(c) Adult-b w/ LR



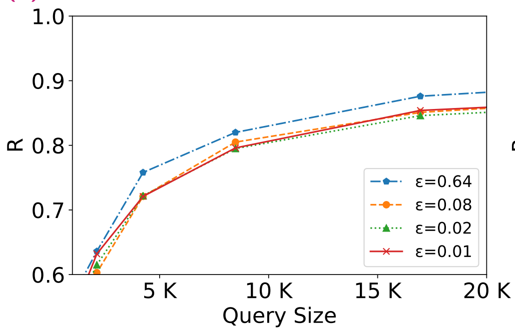
(d) Adult-b w/ NN



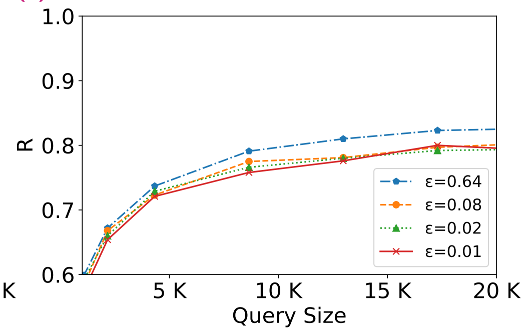
(e) GSS w/ LR



(f) GSS w/ NN

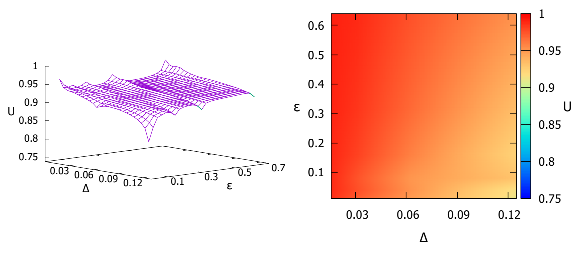


(g) Adult-m w/ LR

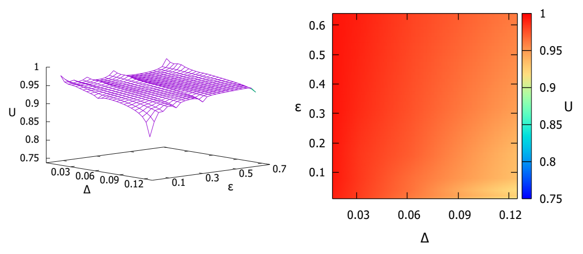


(h) Adult-m w/ NN

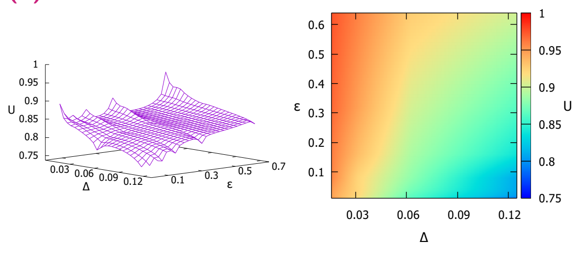
Fig. 3.7: Impact of Varying ϵ in BDPL with $\Delta = 1/8$



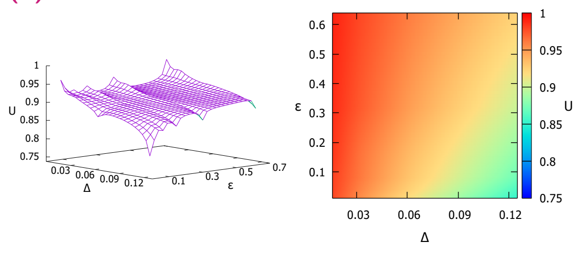
(a) Mushrooms w/ LR



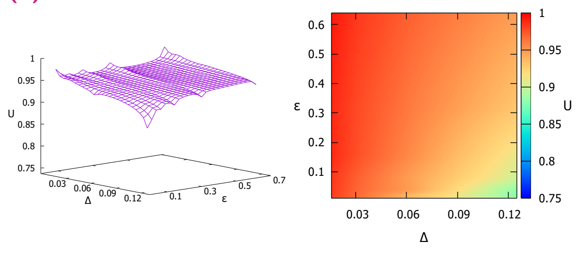
(b) Mushrooms w/ NN



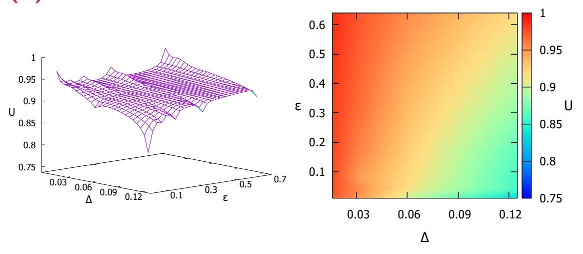
(c) Adult-b w/ LR



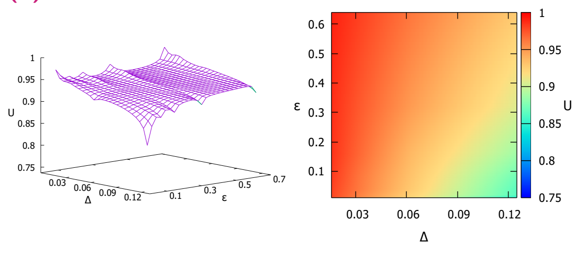
(d) Adult-b w/ NN



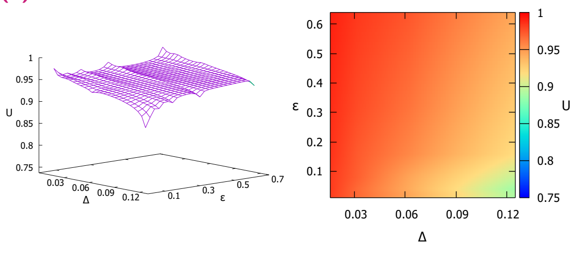
(e) GSS w/ LR



(f) GSS w/ NN

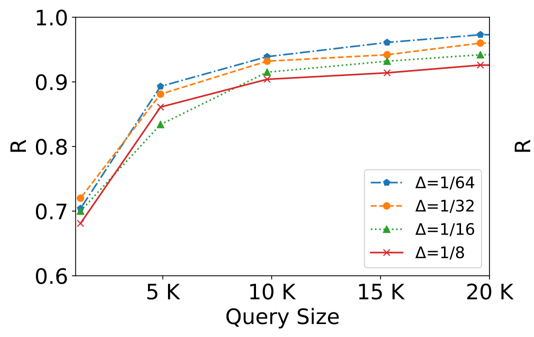


(g) Adult-m w/ LR

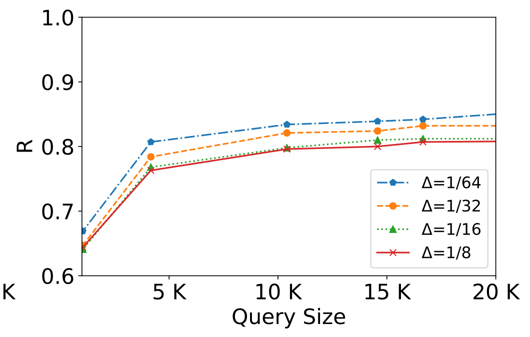


(h) Adult-m w/ NN

Fig. 3.8: Utility vs. Δ and ϵ in BDPL

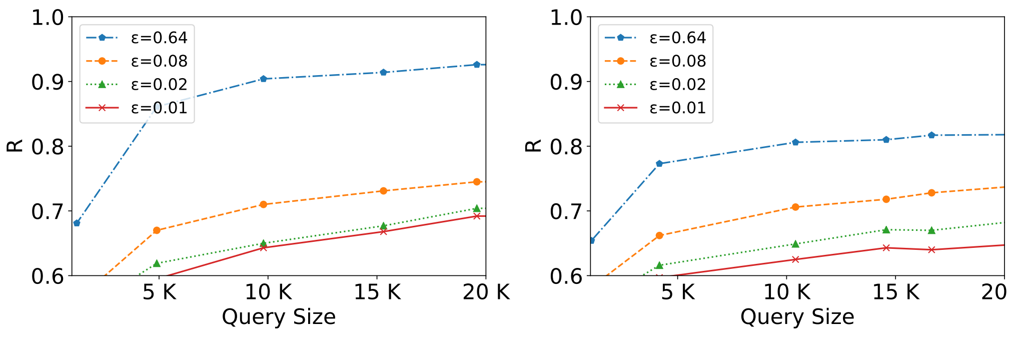


(a) GSS w/ LR



(b) GSS w/ NN

Fig. 3.9: Impact of Varying Δ in Zone-less BDPL



(a) GSS w/ LR **(b)** GSS w/ NN

Fig. 3.10: Impact of Varying ϵ in Zone-less BDPL

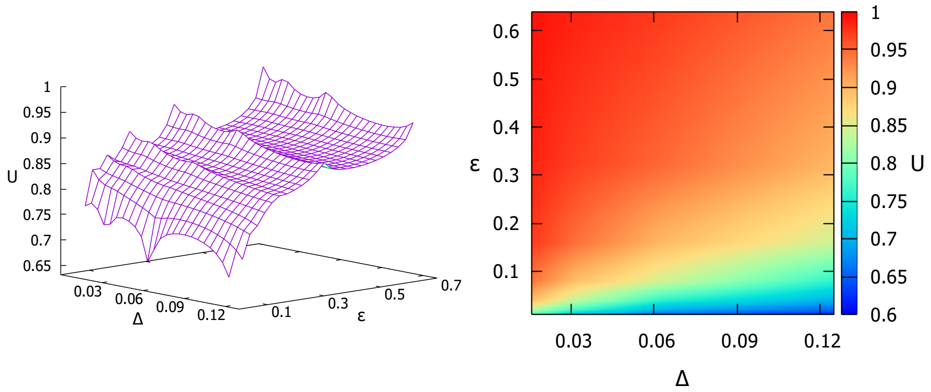
by the fact that zone-less BDPL adopts ABRR which obfuscates results in the global feature space . In addition, utility is still independent of model types and remains over 80% when ϵ is greater than 0.15. As expected, utility reaches peak when $\Delta = 1/64$ (when soft margin is the most concentrated near a decision boundary) and $\epsilon = 0.64$ (the least probability of perturbation).

To conclude, zone-less BDPL provides strong protection for decision boundary in the global feature space. Privacy budget ϵ brings more control over the extraction rate than Δ.

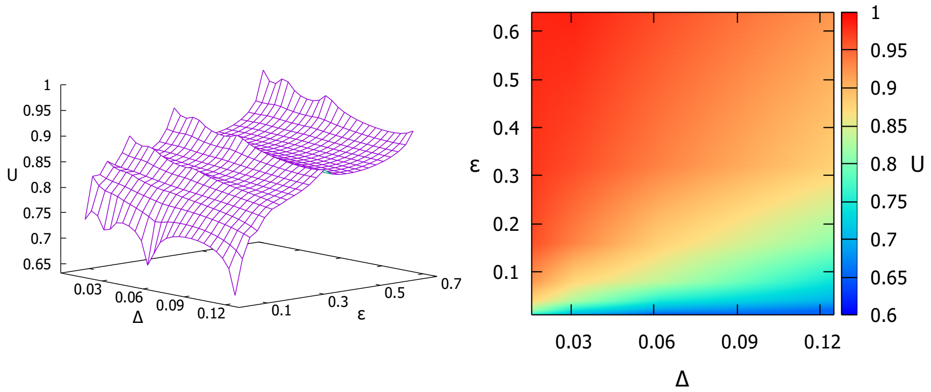
3.3.6 Evaluation of BDPL on Advanced Attack

Recent study has shown that the extraction attacks are becoming threatening on complex models such as convolutional neural network. In this subsection, we turn to these emerging attacks which substantially scale both the input dimensions and model complexity. We expect these to be bigger challenges for BDPL as these attacks allow attackers to draw natural data as query from the same domain such as images.

In Table.3.1, we review those high-quality extraction attacks on complex model from peer-reviewed papers. To precisely address the feature of recent attacks, we list out whether the attacks support two of the most predominant advanced models, i.e., convolutional neural network (CNN) and recurrent neural network (RNN). Adversary knowledge is leveraged to illustrate adversary capability on data acquisition, specifically whether they can access any problem domain dataset. They are divided into three levels with an increasingly stringent requirement on dataset knowledge. We also categorize query strategy based on the nature of query such as reinforce-based probing (e.g., reinforce learning) and adversarial-based probing (e.g., adversarial samples). Detailed techniques can be found in the related works.



(a) GSS w/ LR



(b) GSS w/ NN

Fig. 3.11: Utility vs. Δ and ϵ in Zone-less BDPL

Due to space limitation, we select state-of-the-art ActiveThief[73] as the attack scheme for evaluation because it is the most recent and advanced attack. Activethief is a model extraction framework for neural networks using non-problem domain datasets and pool-based active learning strategies. We adopt the same configuration of the original paper and implement it on a convolutional neural network with various image datasets.

Datasets and Machine Learning Models. We evaluate two datasets for training victim models — a hand-written digits image dataset *MNIST* ($28 * 28$ resolution, 1 channel, $60k$ records) and a colorful general objects dataset *CIFAR10* ($32 * 32$ resolution, 3 channels, $60k$ records). The two datasets are obtained from their official repository respectively [50][48]. Compared to previous evaluation, feature size has scaled to 7x and 30x respectively. As for adversary query database, we use the downsampled and unannotated subset of the ILSVRC2012-14 dataset from ImageNet[20]. In each experiment, images from ImageNet are resized to fit the input size of the victim model.

With regards to the model architecture, we adopt the same CNN design in [73] for evaluation, which has 3 blocks of convolution. In each block, there are 2 repeated units of 2 convolution layers using a 3×3 kernel, followed by 1 pooling layer using 2 × 2 kernel. The stride length is 1 and 2 for convolution kernel and pooling kernel respectively. ReLU is adopted as the activation function for each convolution layer. The last pooling layer is attached to a fully connected layer which produces a final prediction using softmax function.

Attack and Evaluation Metrics. The attack is performed as follows. A random subset of initial seed images are selected from the adversary database. Then the attacker queries these images against the victim model and obtains a set of responses. A basic replica model is developed by training on these query-response pairs. The attacker then queries the remaining images using a designated strategy.

We evaluate one non-probing and one probing strategy regarding decision boundary. The non-probing strategy is ActiveThief Random Strategy (ATRS) where a subset of images are selected uniformly at random. The probing strategy is ActiveThief Hybrid Strategy (ATHS) where k-center and DeeplFool are combined for subset selection and it is the strongest attack in ActiveThief. This process is repeated for a fixed number of iterations. In each iteration, the replica model is retrained from all accumulated query-response pairs. Strategy hyperparameters such as number of seed samples and iteration numbers are the same in [73]. Previous evaluation metrics are adopted, that is, extraction rate R and utility U. To be consistent with original attack, the extraction rate is evaluated against test set samples in the following experiments.

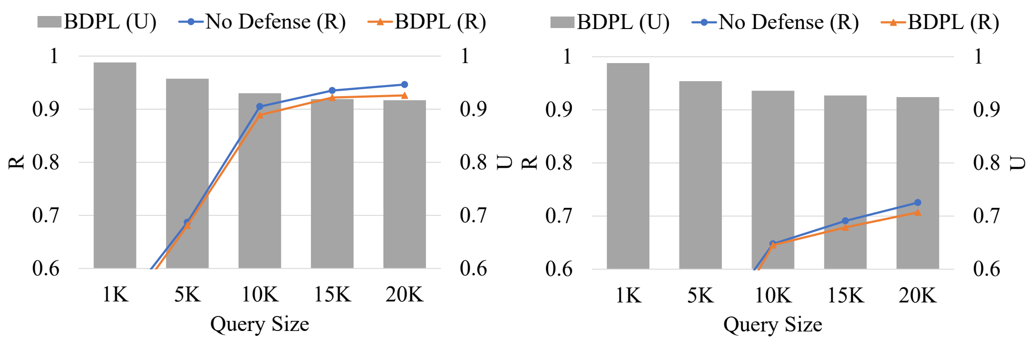
3.3.6.1 Effectiveness of BDPL on Advanced Attack

We launch two extraction attacks (ATRS and ATHS) on 2 models and plot the extraction rate R in Figs. 3.12 and 3.13 in terms of the budget of queries. BDPL parameters $\Delta = 1/7$ and $\epsilon = 0.01$ are set for all models in this experiment.

Effectiveness on ATRS. Fig. 3.12 presents the evaluation results on the non-probing attack. Despite the significant growth of attack complexity, our defense still draws a 1.5% drop over extraction rate on both models. The decrease is small because BDPL focuses queries neighboring decision boundary whereas ATRS draws queries uniformly from normal image distribution and ratios of sensitive queries may be low. The mismatch leads to smaller perturbation as expected. Nevertheless, BDPL still maintains a decreased and saturated upper bound for model extraction.

Table 3.1: Recent Advances in Model Extraction Attacks

Features		Attacks				
		Papernot *et al.* [74]	Juuti *et al.* [41]	Orekondy *et al.* [71]	Yu *et al.* [98]	Pal *et al.* [73]
Victim Model	CNN	✓	✓	✓	✓	✓
	RNN					✓
Adversary Knowledge	Problem Data	✓	✓			
	Non-problem Data			✓		
	Non-problem Data without Labels				✓	✓
Main Query Strategy	Passive Query			✓	✓	✓
	Reinforce-based Probing			✓		
	Adversarial-based Probing	✓	✓		✓	✓
API Output Level	Probability		✓	✓	✓	✓
	Label	✓	✓		✓	✓



(a) MNIST w/ CNN (b) CIFAR-10 w/ CNN

Fig. 3.12: Overall Protection Effect by BDPL on ATRS: Extraction Rate and Utility

From the secondary axis of Fig. 3.12, we observe that the utility trend stabilizes at over 90% after $20K$ queries for both models. Particularly, the utility remains over 95% before $10K$ for *CIFAR-10 w/ CNN*. This is consistent with the small drop of extraction rate given that perturbation is light. The impact on utility by Δ and ϵ will be further discussed in Section 3.3.6.2.

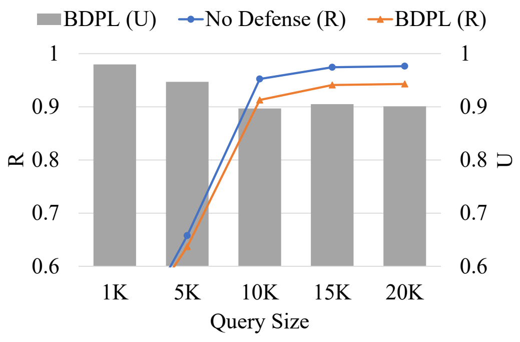
Effectiveness on ATHS. Fig. 3.13 presents the evaluation results on the probing attack. Notably, the extraction is reduced by 4% and 2.7% respectively for two models, which is more significant compared to that in ATRS. This corresponds to the nature of probing strategy that leverages k-center (diversifying classes) and DeepFool (approaching decision boundary). BDPL demonstrates stronger capability against such attack. The drop of extraction rate is smaller in *CIFAR-10 w/ CNN* given that it has great complexity (over 3000 input dimensions) and low risk in current extraction attack.

As for the utility trend, both models are saturated at over 90%. The perturbation stays obviously light which inhibits further drop of extraction rate. We conjecture that current defense is not entirely on its optimal performance due to high-dimensionality and great model complexity. Section 3.4 will further identify the limitations of BDPL and areas of improvement.

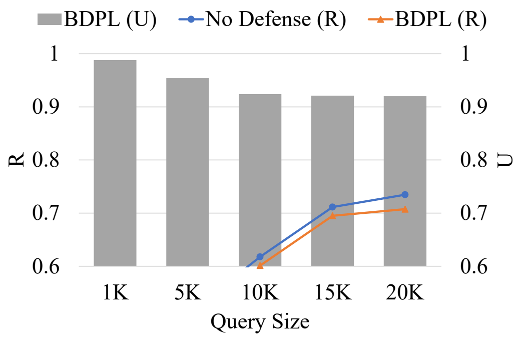
3.3.6.2 Impact of ϵ and Δ on Complex Model

In this subsection, we evaluate BDPL performance with respect to zone parameter Δ and privacy budget ϵ. In Fig. 3.14, we fix ϵ and vary Δ from $1/32$ to $1/4$ in BDPL. In Fig. 3.15, we fix Δ and vary ϵ from 0.01 to 0.64 in BDPL. We mainly plot the results on *MNIST w/ CNN* under non-probing and probing attacks.

The extraction rate is reduced more significantly on ATHS when Δ increases from $1/32$ to $1/4$. As coincided with the design of BDPL, it has bigger impact on probing strategy. Moreover, the drop of extraction rate is obviously

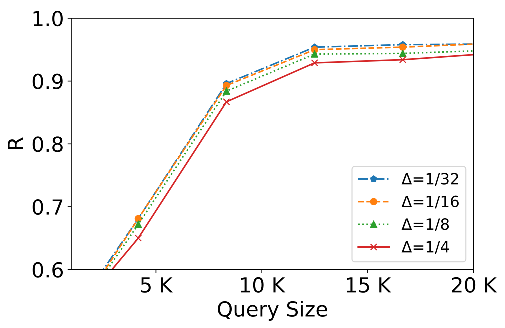


(a) MNIST w/ CNN

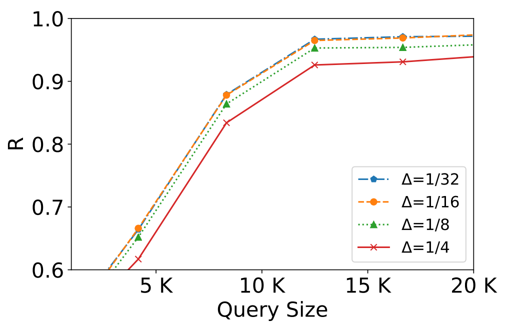


(b) CIFAR-10 w/ CNN

Fig. 3.13: Overall Protection Effect by BDPL on ATHS: Extraction Rate and Utility



(a) ATRS



(b) ATHS

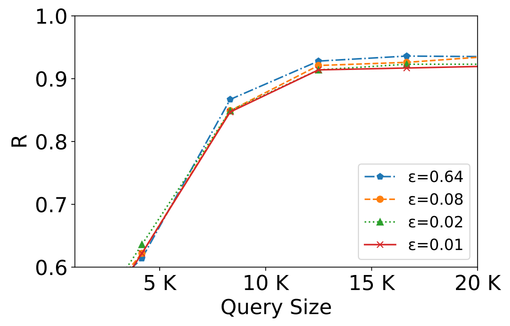
Fig. 3.14: Impact of Varying Δ in BDPL on MNIST w/ CNN

greater on $\Delta = 1/4$ than the other 3 parameters, which indicates that a large Δ is required for effective defense on the current model. As for ϵ, when decreasing to 0.01, the change is less significant compared to that in Δ. This indicates that complex model is less sensitive against the change of perturbation (privacy budget). Overall, the extraction rates are consistently bounded and saturate as the query size increases. ϵ-BDPL still imposes a theoretical upper bound on the extraction rate and prevent full extraction. Furthermore, BDPL displays flexible control over the extraction in probing strategy.

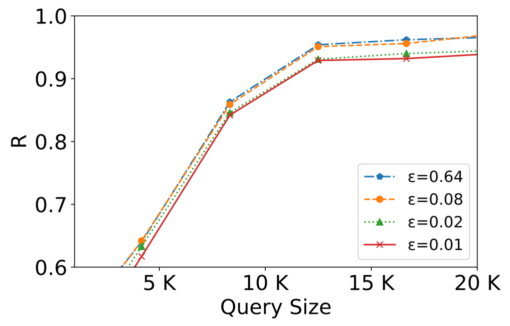
To conclude, BDPL alleviates the threat of extraction rate in spite of the significant growth of victim complexity and strong attack. The change of Δ takes dominant control over the extraction rate. We will discuss the limitations and improvements in the following section.

3.4 Discussion

In this section, we make some notes of BDPL and identify areas of future improvement.



(a) ATRS



(b) ATHS

Fig. 3.15: Impact of Varying ϵ in BDPL on MNIST w/ CNN

We set $\Delta = 1/8$ and $\epsilon = 0.01$ for the demonstration of overall evaluation as this combination can strike a good balance between the security and the utility while showing transferability among datasets. However, for the practitioners, this combination may not always be the best choice during deployment. They will need to adjust Δ and ϵ sequentially according to their demand and the flexibility plot in this thesis. To improve the deployment efficiency in future work, a self-adaptive mechanism for hyperparameter is needed to set Δ, ϵ given the security and utility goal. It can initialize the layer using the above combination and then adjust them on the fly. The security level can be measured using other monitoring-based techniques such as feature coverage [46] while the utility can be estimated through the perturbation mechanism.

For attacks that only support probability-level extraction, our BDPL, which is at label-level, cannot protect against it. On the other hand, we'd argue that BDPL can still protect against attacks extraction using natural data. First, normal and natural data can also be close to decision boundary, although the ratio of sensitive queries in these attacks is lower than that in the fine-tuned ones. Second, the optimal strategy in recent studies [41], [73] leverages adversarial techniques which implicitly perform fine-tuned probing on the decision boundary. This means our BDPL can effectively protect against them, as indicated in our new experimental results against [73] in Section 5.6. Nonetheless, extraction using natural data is limited on specific model types, such as images and text, where same domain data can be easily obtained and used as query set. If the victim is a genetic model, it would be difficult to perform such extraction since genetic data is usually proprietary.

We also identify two core components that can be further improved for better performance in complex model. One is distance metrics and the other is perturbation mechanism.

Distance Metrics. In the current version, we adopt flipping-corner-point technique on two essential assumptions: modular design and generality. The first one allows plug-in feature for practical deployment, where service providers can tap in BDPL with the same API for users. The second one ensures compatibility with both parametric (e.g. neural network) and non-parametric models (e.g., decision tree). Unlike the classical models, deep neural networks such as image models may have high model complexity and thousands of input dimensions. Our intuition for flipping-corner technique, which comes from the L-norm ball, may have unexpected behavior in such space[3].

To improve the scalability, we can start by relaxing the first assumption and access the internals of provider's model. As such, a straightforward remedy becomes feasible by performing flipping-corner detection in the middle part of model, such as a bottleneck layer[79]. The dimensionality is greatly reduced after the intermediate representation compared to the raw input. Apart from the high-dimensionality, the manifold assumption discussed in adversarial robustness of complex model may also render flipping-corner-point technique unstable. Corner points may not flip properly when samples fall out of the manifold. As a result, L-norm distance metrics may degrade the accuracy of detection process. By relaxing the second assumption, we can leverage the gradient from parametric model to propose gradient-based distance metrics, which is more suitable under the manifold assumption. We believe it is a viable and practical solution for future improvement as complex parametric models are prevailing in machine learning services.

Perturbation Mechanism. As motivated by binary defense, randomized response is adopted as the basic framework for multiclass defense. We perform a one-vs-one approximation for each class and treat multiclass defense as a combination of binary defense. This assumption may incur imbalanced noises since only one counter class is considered. Perturbation may be concentrated on specific pairs of classes. To resolve this, a natural framework capable of categorical-value perturbation, such as k-ary randomized response[42], can be adopted for multiclass defense. Furthermore, if we extend the mechanism to be numeric-value suitable, probability-level defense becomes feasible. Nonetheless, these existing mechanisms will still need significant adaptation to satisfy ϵ-BDP before applying it to our defense. We plan to implement them in future work.

3.5 Summary

In this chapter, we propose boundary differentially private layer to defend machine learning models against extraction attacks by obfuscating the query responses. This layer guarantees boundary differential privacy in a user-specified boundary-sensitive zone. To identify sensitive queries that fall in a zone, we develop an efficient approach that uses corner points as indicators. We design boundary randomized response as the building block for perturbation algorithm, followed by a generalization to multiclass model and an adaptive version that can protect a soft margin of decision boundary. We prove such perturbation algorithm satisfies ϵ-BDP. Through extensive experimental results, we demonstrate the effectiveness and flexibility of our defense layer on protecting decision boundary while retaining high utility of the machine learning service. For future work, we plan to propose defense more suitable for complex model and consider strong adversary with evasion. We also plan to extend our defense layer to protect against other machine learning attacks such as model evasion and inversion.

Learning Indoor Locations from Unprivileged Sensor Data

4

In this chapter, we turn to the attack surface of data pool and investigate learning-based inference on unprivileged sensor data. The increasing sensory capability and accuracy in mobile and wearable devices have nourished many convenient applications, such as turn-by-turn navigation, fitness tracking, virtual reality, and interactive mobile game. However, privacy infringement arising from these applications has recently drawn much attention throughout the world. Unfortunately, as more and more personal data, such as locations, passwords and daily schedules, are accessed through smartphones, even the best practice of privacy protection cannot protect them against mobile attacks that exploit side-channel information, such as UI state [12], power usage [63], or cellular network signal strength [83].

In the literature of side-channel attacks, many works have succeeded in exposing information about victim's location such as tracking their driving or public transport routes without using GPS, either through cellular/Wi-Fi networks [83] or by inertial sensors [36], [66]. However, these works focus on outdoor location, so it remains unresolved on the risk of **indoor** location leakage from unprivileged sensory data, which are usually more private and sensitive. In this chapter, we develop the **m**obile **i**nertial **s**ensor-based **s**ensitive **i**ndoor **l**ocation **e**avesdropping (**MISSILE**) system to infer sensitive indoor locations using side-channel information only from unprivileged sensors such as accelerometer, gyroscope and magnetic field sensor. Our key idea is to identify a sensitive indoor location (e.g., an office) using multiple structural characteristics (e.g., turnings in a corridor, pausing of motion to open a fire stop door, or taking an elevator). These characteristics lead to unique patterns in sensor readings and constitute the signature of this location.

There are four challenges in MISSILE, namely, how to acquire reliable location labels, how to handle data inconsistency caused by device placement and movement, how to transform raw data into features, and how to build an effective learning model. To address these challenges, we propose a general-purpose machine learning system without prior knowledge of structural characteristics. To feed this system with sufficient training data, we develop an automatic location labeling mechanism using Bluetooth beacons with latency calibration method. Raw sensory data collected from different sources are made consistent with normalization and noise reduction techniques. After an efficient feature extraction procedure, calibration for anomalies is further applied in modeling to reduce the impact of data contamination from mislabeling and low-quality sensor output. Finally, a lightweight classifier is trained and embedded in a spyware to eavesdrop a victim's sensitive indoor location. Through our extensive experiments in a real indoor environment, we show the feasibility of MISSILE and the high risk of indoor sensitive location eavesdropping. To complement this research, we also discuss the potential extension of this attack and two countermeasures in addition to lifting up the privilege requirement of accessing sensory data.

To summarize, our contributions of this study are in the following three perspectives.

- We adopt a general adversarial framework for side-channel attacks on mobile devices, based on which we propose our indoor location eavesdropping attack.
- We develop a real-life indoor location eavesdropping attack system which comprises automatic data labeling, data processing and machine learning pipeline on mobile inertial sensor data.
- We propose a labeling mechanism with BLE beacons and a calibration method to compensate for latency using maximum likelihood estimation.
- We conduct extensive experiments to demonstrate the feasibility of such an attack and thus the risk of indoor location exposure in practice.

The rest of this chapter is organized as follows. Chapter 4.1 formally defines the privacy problem from side-channel attack and threat model with challenges. Chapter 4.2 dives into the detail of MISSILE system and its associated algorithms. Chapter 4.3 presents the system evaluation and severity of this threat. We further discuss the extension of this system and potential countermeasures in Chapter 4.4. Finally we draw our summary of this study in Chapter 4.5.

4.1 Problem Statement

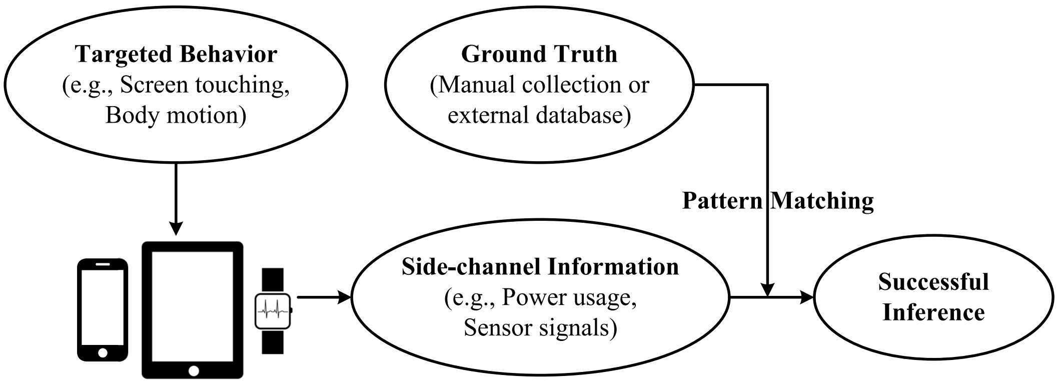

Targeted Behavior
(e.g., Screen touching, Body motion)
Ground Truth
(Manual collection or external database)
Pattern Matching
Side-channel Information
(e.g., Power usage, Sensor signals)
Successful Inference

Fig. 4.1: The framework for side-channel attack on mobile devices

4.1.1 General Side-channel Attack Framework

A typical side-channel attack on mobile devices is described in Fig. 4.1. Side channels in these devices may react to user interaction or exterior environment change, which can be exploited by attackers to infer sensitive information. For example, a slight but distinct acceleration change in motion sensor can leak a user's keystroke on soft keypad. A direct consequence of such attack is the loss of users' privacy, which may further lead to even more serious attacks such as social engineering on the victim, blackmailing ransomware, and hijacking. As most sensors (especially multiple inertial sensors) do not require permission to access, such attacks can be camouflaged in normal applications, which makes them hard to detect. Based on this general adversarial framework, in what follows we define the MISSILE attack on indoor sensitive locations.

4.1.2 MISSILE Motivating Scenario

We assume there are a finite number of sensitive locations within the premises concerned (such as a campus, a shopping center, or a hospital). An adversary would like to stalk the daily routine of a frequent visitor (such as students/staff in a university campus) and to eavesdrop whether and how often a victim user visits some sensitive location such as an office room, a particular clinic, or even restroom. We assume the adversary can intrigue the victim to install a legitimate application on her mobile phone.[1] Victim users are often tricked into downloading such apps especially when they do not require special permissions such as location. For example, Kaspersky Labs found and removed 58,000 instances of stalkerware in 2018 [31]. Even

[1]Some studies have also revealed the possibility of attaining sensor data through web browsers using Javascript [61], notwithstanding limited sensor types (e.g., motion sensors only) and sampling frequency.

popular “trustworthy” apps might have vulnerabilities that open the door for spying, such as the one found in Whatsapp that allows injection of spyware onto people’s phones [68]. Through such applications, the adversary can then collect unprivileged sensory information on the victim’s mobile device in the background (both Android and iOS allow such collection without permission). A classifier embedded in this application can then identify the unique sensor pattern of a sensitive location.

The sensors considered in this chapter include accelerometer, linear acceleration sensor, gyroscope, and magnetic field sensor. While accelerometer and linear acceleration sensor are common motion sensors shipped in modern mobile devices for detecting device acceleration, gyroscope is another important sensor. By detecting a sudden turn or a subtle slow winding, it indicates if a victim user is changing his/her direction in a regular degree due to a hallway or corridor. Magnetic field sensor is an environmental sensor whose readings change as the victim user moves indoor. Magnetic local variation exists in all buildings due to the geolocation and magnetic materials used in construction (e.g., a large amount of steel in an elevator) [53]. By combining the above sensor readings in mobile devices, each sensitive location may have a unique pattern in the sensory data stream for location inference.

4.1.3 Threat Model

The major threat comes from a mobile application that only silently collects sensory data and eavesdrops sensitive location. In this chapter, we assume the attacker and its client-side application has the following capabilities or characteristics:

Adversary Application and Network: For both Android and iOS, application packages from any sources can be installed on the devices.[2] As such, the malicious party can easily develop legitimate spyware or repackage popular applications (such as Facebook, Messenger, and WhatsApp) with malicious codes and distribute them through social networks, third-party app markets or emails. We assume this application has network access, either Wi-Fi or cellular network, to upload the eavesdropping results to or update the classifier regularly from the attacker’s server.

Stealthy Side Channels: Side channels obtained from the unprivileged accelerometer, gyroscope, and magnetic field sensor are accessible to the adversary application. While both Android and iOS have permission pro-

[2]Apple Developer Enterprise Program allows a developer to create and distribute custom apps to any iOS device without submitting them to App Store.

tection mechanism for GPS, Wi-Fi and Bluetooth (iOS and Android over 6.0 require on-the-fly approval), there is no specific permission protection for sensors on both operating systems. The attack is not assumed to be zero-permission. Instead, since the attack only targets at the permission-free inertial sensors, no additional permissions (particularly location-related) are needed. In other words, any installed application can acquire sensor readings without the consent or even knowledge of users. Existing antivirus apps cannot prevent MISSILE from running in the background as MISSILE only monitors sensor readings with low CPU and battery consumption like most legitimate applications.

Computational Power and Machine Intelligent: The application can access the CPU (or even GPU) of the mobile device for sensory data processing and classification. Nonetheless, the computationally intensive training of the classifier is still performed on the server side. However, as the computational capability of mobile devices keeps increasing, especially with the advent of dedicated AI chip on SoC (e.g., ARM Machine Learning Processor), certain machine learning tasks can be processed on the mobile devices to offload the MISSILE server and improve location inference response time.

4.1.4 Technical Challenges

Indoor pedestrian location inference using sensory data is more challenging than route inference in outdoor environments [66], [95]. We summarize four major challenges as below.

Reliable Label Acquisition: To perform indoor location eavesdropping attack, we need to capture sensor readings with proper labels. As GPS and open map data are usually not available under indoor scenarios, an automatic, highly-efficient, and reliable mechanism is needed to collect a large number of location labels as ground truth for training data.

Data Inconsistency: Since the output coordinates of inertial sensors depend on the relative posture of mobile devices, we need to normalize various device placements such as vertically in a pocket or horizontally in a handbag. Furthermore, motion sensors capture not only the location pattern but also the walking style of users. The diversity of walking speed and moving behavior of individuals has a negative impact on the inference as it causes inconsistency in sensory data.

Raw Data Optimization: Raw sensor values are not suitable to be directly fed into a machine learning pipeline since processing high-dimensional and high-frequency data consumes a significant amount of computational resources. To maximize the attack performance under limited computational power, we need an optimal set of low-dimensional features selected by an automated feature extraction procedure without prior information of location details.

Robust Modeling: The collected training data may be in low quality, as they can be contaminated by corrupted devices, label signal delay, and internal software or device faults. The performance of the machine learning model can be impacted by such anomalies and therefore a robust model with anomaly calibration is always preferred.

4.2 MISSILE System

In this section, we first present the overall design of MISSILE system, followed by the detailed discussion on individual component implementation.

4.2.1 Design Overview

As shown in Fig. 4.2, the proposed MISSILE system is composed of two stages.

In the training stage, the attacker first identifies target indoor sensitive locations, physically walks through these locations with stock mobile devices, and collects sensor readings as they pass by these locations.[3] To automate the collection process and increase its accuracy, MISSILE deploys a Bluetooth Low Energy (BLE) beacon in each sensitive location to activate sensor readings automatically as the attacker walks by. BLE beacons (e.g. Apple's iBeacon) are small, inexpensive, and long-lasting devices that continuously emit identifiable radio signals in the neighborhood (normally within a range of up to 10 meters in our system). In practice, BLE beacons have been widely deployed by many indoor positioning services for navigation and advertisement, so the attacker can even leverage these existing beacons without any extra deployment cost.

To acquire a desired length of data with proper label, **segmentation** is performed on the long continuous data stream. Such a small segment from the whole stream is called an **exemplar**, which is assumed to contain the unique signature of a sensitive location. The length of exemplar is a

[3]Many premises are semi-private/semi-public and accessible to the attacker. For example, everyone can enter most of the buildings in a university campus or a hospital even though they are privately owned premises.

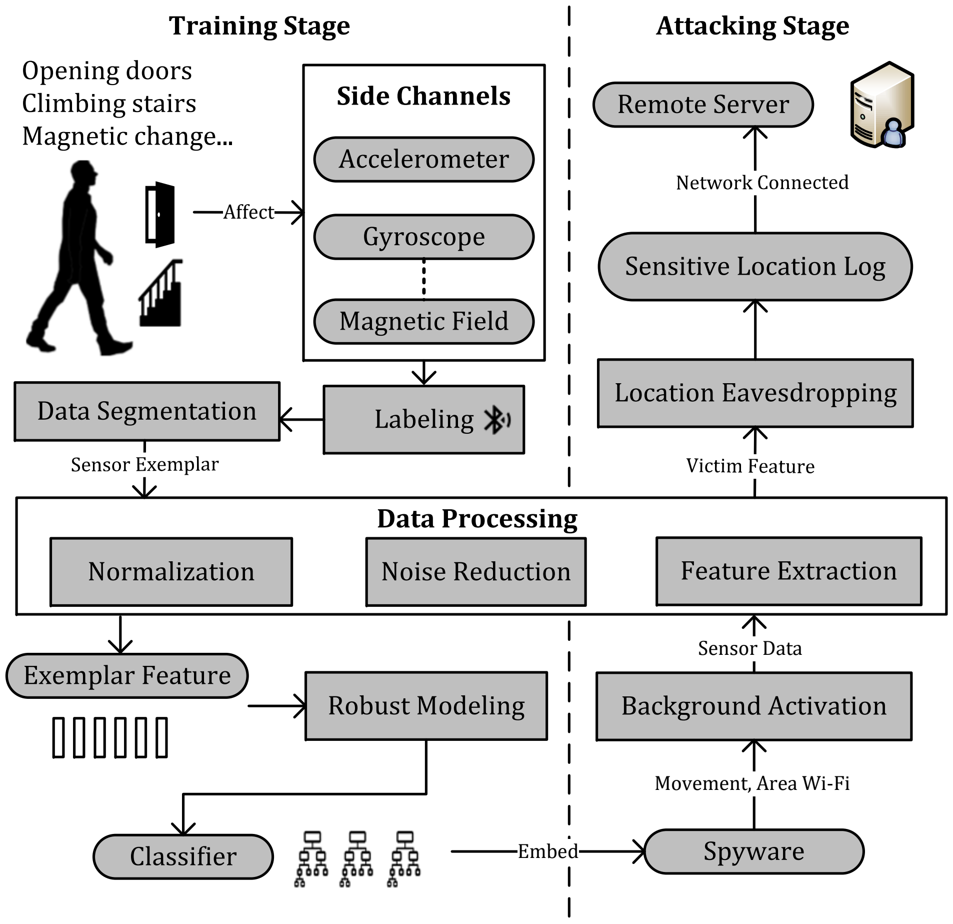


Fig. 4.2: Overview of MISSILE system

hyperparameter that ensures it is long enough to contain the desired sensor data patterns. Each exemplar can contain signals from multiple sensors to capture a comprehensive set of location characteristics, so that they can reveal the structural (e.g., door opening, stairs walking), ambient (e.g., magnetic field), and even environmental (e.g., air pressure) patterns and increase robustness against dynamic environment such as high user density.

Exemplars are further normalized to resolve the inconsistency problem of device placement. Noise reduction is applied next due to the high-density noise in normalized exemplar from body movement. Then automatic feature extraction is performed to obtain an optimal low-dimensional representation of the sensor pattern. The generality of this procedure allows the attacker to replicate MISSILE to other premises without knowledge of the actual sensor pattern and feature engineering. When the features of clean exemplars are ready, a robust supervised learning scheme using anomaly calibration technique is used to construct a classifier to recognize the sensor pattern for each sensitive location.

In the attacking stage, the attacker embeds this classifier into a legitimate mobile application for victims to install on their mobile devices. This application then continuously collects the sensor readings in the background and captures indoor sensitive locations when the expected sensor patterns occur. To preserve battery life, two activation techniques are introduced to reduce unnecessary eavesdropping when the victim is far from the concerned premises or is stationary. Finally, the eavesdropped sensitive location log can be delivered to the attacker when the network is available.

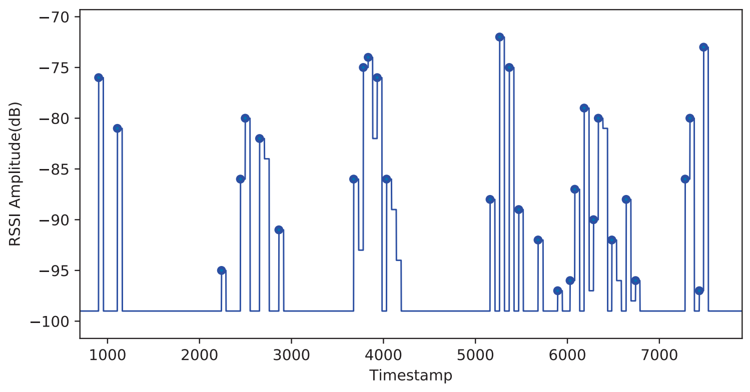
4.2.2 Labeling and Data Segmentation

The first key component is to segment the short, recognizable pattern exemplar of sensitive locations from the stream of continuous sensory readings. To determine the starting timestamp of an exemplar, an intuitive choice is to use the estimated distance from the beacon. However, since this distance is hard and inaccurate to estimate,[4] we instead use the raw Received Signal Strength Indicator (RSSI) and its change. Typically, RSSI ranges from around -20dB to -80dB in short proximity and less than -95dB in the farthest distance under the setting of experiment deployment.

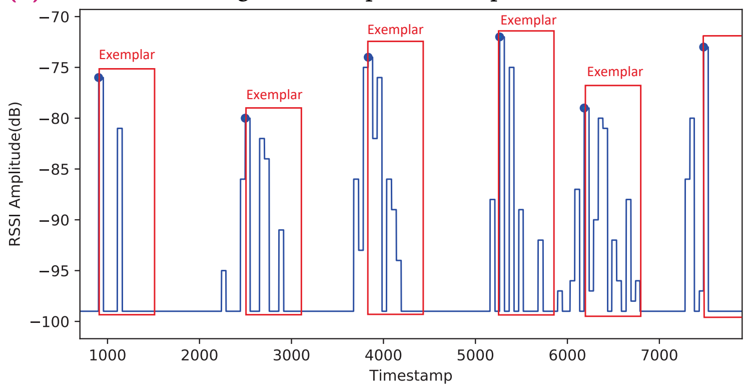
4.2.2.1 Climbing Point as Starting Timestamp

Since this is the training stage, the attacker can have the full control to keep the device moving while collecting the BLE signals. The challenge in segmentation is to determine the *starting* timestamp of a potential sensor pattern that indicates a sensitive location is reached. Intuitively, this timestamp should be associated with a maximal RSSI value (i.e., a climbing point). However, due to the fluctuation of radio signals, there are multiple climbing points when walking through a location, as illustrated in Fig. 4.3a. To resolve the true starting timestamp, we introduce two thresholds to prune climbing point candidates caused by signal delay and other factors. *Step threshold* is the minimum length between two starting timestamps (of two locations), and the RSSI threshold defines the minimum RSSI for a starting timestamp. The former is based on the fact that sensitive locations are discrete and fall apart with one another, whereas the latter is based on the fact that the starting timestamp is usually associated with a strong RSSI. When multiple reference points are available in a location, we leverage the metadata emitted

[4]Theoretically, we can estimate the distance between a receiver and a beacon based on the received signal and the reference signal strength of 1-meter distance. However, due to the environmental absorption and power change, such distance estimation can suffer from significant delay and fluctuations.



(a) Potential starting timestamps of exemplar



(b) Selected starting timestamps with st = 750 and rt = -85dB

Fig. 4.3: Exemplar timestamp is detected in RSSI sequence collected from one reference point. The sensitivity is set to -99dB since RSSI lower than this threshold occurs from a remote beacon.

by the beacons to separate signal sources, namely major identifier and minor identifier. In our setting, major identifier denotes the location while minor identifier denotes the beacon itself.

Algorithm 4 describes the details of determining starting timestamp for an exemplar. It first separates the RSSI sequence from different beacons by $minor$. After deriving all RSSI climbing points from each beacon, we store them in the set of CP as shown in Fig. 4.3a. A point is defined as *climbing* if the current RSSI is larger than its previous moment in the sub-sequence. Based on the provided RSSI threshold rt, all climbing points whose RSSI values are below rt are pruned. The algorithm iteratively sorts and accesses remaining points in descending order of their RSSI values. In each iteration, only one climbing point is retained for each beacon within the step threshold st while all other climbing points with different $major$ (i.e., signals from other locations) are pruned. After this step, only those strong climbing points survive in $Start$, the candidate set for starting timestamps of sensor pattern. Fig. 4.3b illustrates a running example of this algorithm for one reference point (i.e., RSSI measurements taken from one beacon for each

Algorithm 4 Starting Timestamp Determination

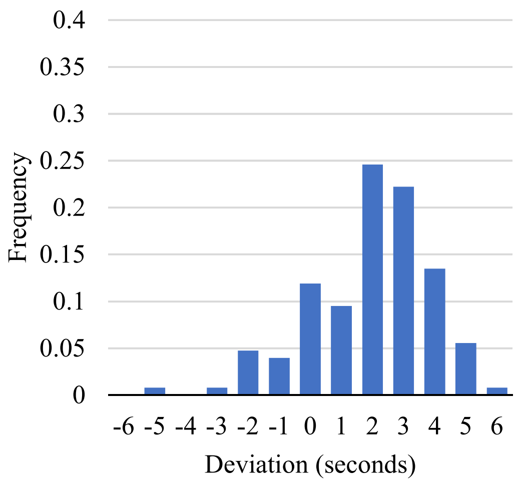
```
Input:      RSSI sequence
            $S = \{s_1, s_2, \ldots, s_n\}$
            $s_i = \{timestamp, major, minor, rssi\}$
            Step threshold $st$
            RSSI threshold $rt$
Output:     Starting points $Start$
Procedure:
 1: $CP = \emptyset$, $Pruned = \emptyset$, $Start = \emptyset$
 2: Segment $S$ into sub-sequences $\{S_1, S_2, \ldots\}$ with same $minor$
 3: for each sequence $S_i$ do
 4:     Add all climbing points in $S_i$ to $CP$
 5: for $j = 1$ to $|CP|$ do
 6:     if $CP_j.rssi \leq rt$ then
 7:         Prune $j$-th point from $CP$
 8: $CP = DescendSortRSSI(CP)$
 9: for $k = 1$ to $|CP|$ do
10:     if $CP_k$ not in $Pruned$ then
11:         $l = CP_k.timestamp - st$
12:         $r = CP_k.timestamp + st$
13:         $O = FindOverlap(l, r, CP, CP_k.major)$
14:         $Pruned = Pruned \cup O$
15: $Start = CP - Pruned$
16: Return $Start$
```

location), where a red rectangle denotes an exemplar of length 15 seconds (i.e., 750 samples under a 50Hz sampling rate). In what follows, we propose a calibration method to refine this starting timestamp to further compensate for the latency of BLE signal.

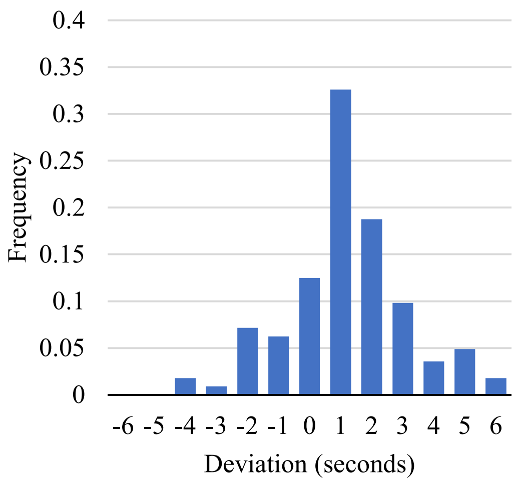
4.2.2.2 Calibration for Latency

The latency of detecting BLE beacon signal consists of both discovery latency and propagation latency. The former arises from the fact that BLE is a slotted protocol that periodically sends data packet in designated time slots and sleeps in between. The emitting interval between two consecutive slots can range from 100ms to 2000ms. Discovery latency happens when broadcast packets miss the scanning window of a receiving mobile device, which has low BLE scanning frequency by default. Since data collection is managed by attacker, such latency can be significantly reduced by minimizing the emitting interval and maximizing the scanning frequency [15].

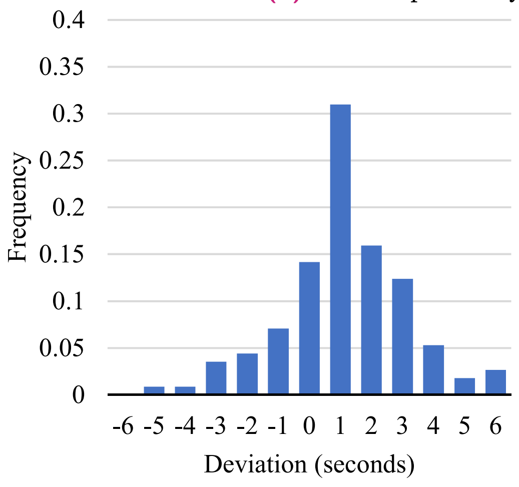
The propagation latency is caused by radio signal propagation due to absorption, congestion or reflection. Although such latency could be large and fluctuating in general, we only care about the latency when the device is in close proximity to the BLE beacon to annotate the starting timestamp.



(a) 2 meter proximity of Beacon A



(b) 1 meter proximity of Beacon A



(c) 1 meter proximity of Beacon B

Fig. 4.4: The deviation histogram of derived starting timestamp

As shown in Fig. 4.4, we plot the deviation between the actual starting timestamp, which is recorded manually, and the derived starting timestamp from Algorithm 4 under different proximity distances and beacon models. We observe that the deviation can be approximated by Gaussian distribution with a mean proportional to its proximity distance. Under this assumption, we propose a calibration method using *Maximum Likelihood Estimation* (MLE) [94] as follows to refine the annotated starting timestamp.

According to *Bayes' Theorem*, the conditional probability of actual starting timestamp t given a derived starting timestamp α from RSSI is

$$P(t|\alpha) = \frac{P(\alpha|t)P(t)}{P(\alpha)}.$$

Since both P(α) and P(t) are constant (because both α and t are uniformly distributed drawn from their domains), maximizing $P(t|\alpha)$ is equivalent to maximizing P($\alpha|t$), and further P($\alpha - t|t$), the conditional probability of deviation $\alpha - t$. According to our assumption, the latter follows a Gaussian distribution, i.e., $P(\alpha - t|t) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(\alpha-t-\mu)^2}{2\sigma^2}}$. Therefore, we can calibrate the starting timestamp t^* by maximizing the following likelihood

$$\begin{aligned} t^* &= \arg\max_t \, P(\alpha - t|t) \\ &= \arg\max_t \, \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(\alpha-t-\mu)^2}{2\sigma^2}} \\ &= \arg\min_t \frac{(\alpha - t - \mu)^2}{2\sigma^2}. \end{aligned} \tag{4.1}$$

After solving the Eqn. 4.1, we have

$$t^* = \alpha - \mu. \tag{4.2}$$

According to Eqn. 4.2, in the single reference point case, the calibration can simply be carried out by deducting a mean deviation from the derived starting timestamp.

Now we generalize the derivation to the case of two reference points (e.g., beacons on both sides of the location) whose derived timestamps are α_1 and α_2 respectively.[5] The joint conditional likelihood of α_1 and α_2 can be derived from their individual distribution independently:

$$P(\alpha_1, \alpha_2|t) = P(\alpha_1|t) \cdot P(\alpha_2|t).$$

Similar to the single reference point case, we can calibrate the starting timestamp t^* by maximizing the joint likelihood of $\alpha_1 - t$ and $\alpha_2 - t$ instead:

$$\begin{aligned} t^* &= \arg\max_t \, P(\alpha_1 - t|t) \cdot P(\alpha_2 - t|t) \\ &= \arg\max_t \, \frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(\alpha_1-t-\mu_1)^2}{2{\sigma_1}^2}} \cdot \frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(\alpha_2-t-\mu_2)^2}{2{\sigma_2}^2}} \\ &= \arg\min_t \, \frac{(\alpha_1 - t - \mu_1)^2}{2{\sigma_1}^2} + \frac{(\alpha_2 - t - \mu_2)^2}{2{\sigma_2}^2}. \end{aligned} \tag{4.3}$$

As such, by solving Eqn. 4.3, we have

$$t^* = \frac{{\sigma_2}^2(\alpha_1 - \mu_1) + {\sigma_1}^2(\alpha_2 - \mu_2)}{{\sigma_1}^2 + {\sigma_2}^2}.$$

[5]We assume no collision in the beacon signal as BLE can transmit through 40 channels.

As we observe from Fig. 4.4, the deviation follows the same distribution given the same beacon model and proximity distance. Therefore, we can simplify t^* by setting $\sigma_1 = \sigma_2 = \sigma$ and $\mu_1 = \mu_2 = \mu$:

$$t^* = \frac{\sigma^2(\alpha_1 - \mu) + \sigma^2(\alpha_2 - \mu)}{\sigma^2 + \sigma^2} = \frac{\alpha_1 + \alpha_2}{2} - \mu. \tag{4.4}$$

Eqn. 4.4 means that in case of two or more reference points, the calibration can simply be carried out by deducting a mean deviation from the average of all derived starting timestamps.

4.2.3 Normalization and Noise Reduction

Most inertial sensors (e.g., accelerometer) produce 3-dimensional readings in a coordinate system that is relative to the device's screen. As such, different device placements cause inconsistency of the sensor readings even when they come from the same location. Another key factor in data consistency is irrelevant noise caused by body movement. For example, walking has a major impact on motion sensors especially when the device is placed close to the leg (e.g., in the pant pocket). In such cases, the sensor signals caused by body movement can overshadow those caused by the physical environment.

4.2.3.1 Resolving Inconsistency by Device Placement

A straightforward solution is to convert the screen-based 3-axis coordinate vector, such as the accelerometer vector $A = [a_x, a_y, a_z]$, into an absolute value by taking the Euclidean norm:

$$\|A\| = \sqrt{a_x^2 + a_y^2 + a_z^2}.$$

This scalar is independent of device placement, but the details of device movement on each axis are removed. To preserve the details, we adopt the rotation-based normalization which transforms screen-based coordinate into world reference coordinate [29]. In what follows, we use the gravity sensor vector and the magnetic field sensor vector as example. Note that the former points to the core of the earth while the latter always provides an approximate geographical pole direction. A rotation matrix which maps between the screen-based coordinate and world coordinate can be derived as follows.

Unit vector of a vector v can be obtained from $v_u = \frac{v}{\|v\|}$. Let G and M be the unit vector of gravity and magnetic field in device reference, the cross

Table 4.1: Runtime on Google Pixel

Approach	Execution Time (ns)
Coordinate Rotation	20000 - 25000
Euclidean Norm	600 - 900

product of G and M must be perpendicular to the plane spanned by G and M. Since M lies on the plane spanned by the gravity vector and south-north vector, this cross product produces vector EW, i.e., the west-east vector. Similarly, the south-north vector SN is the cross product of G and unit vector of EW:

$$G = [g_x, g_y, g_z]^\top, \quad M = [m_x, m_y, m_z]^\top,$$

$$M \times G = EW, \quad G \times EW = SN.$$

As such, we can use the unit vectors of EW, SN, G to form a rotation matrix that connects screen-based coordinate and world reference coordinate. To rotate a new sample K into world coordinate, we multiply it with the inverse of rotation matrix R as follows

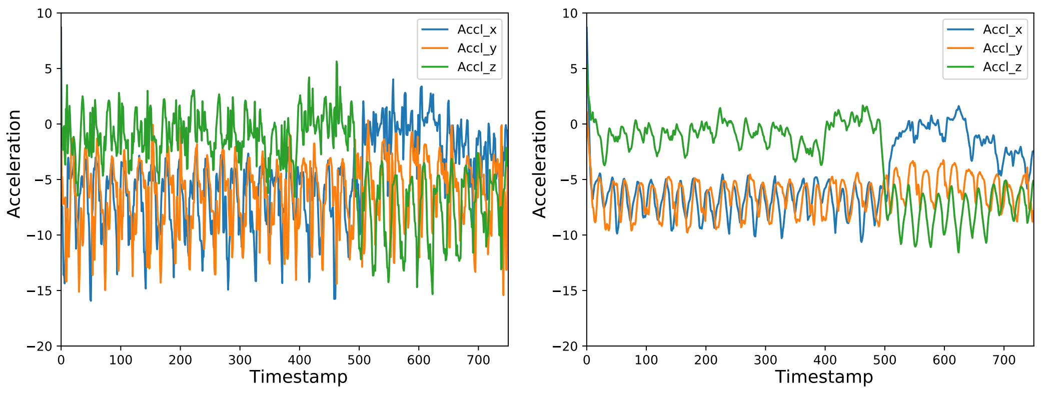
$$R = \begin{bmatrix} ew_x & sn_x & g_x \\ ew_y & sn_y & g_y \\ ew_z & sn_z & g_z \end{bmatrix}.$$

$$R^{-1} \cdot K = K_{rotated}.$$

Obviously the computational cost of the rotation-matrix-based normalization is higher than the Euclidean-norm-based normalization, as the former involves matrix inverse and multiplication. In our experiment, we measure their CPU time (see Table 4.1), and the latter is more than 20 times faster. Nonetheless, the former preserves more details in each axis and our experimental results in Table 4.5 show that the former consistently outperforms the latter in terms of F1-score under various classifiers.

4.2.3.2 Resolving Inconsistency by Body Movement Noises

Body movement noises are mostly distributed in the high-frequency spectrum while sensor signals corresponding to location patterns lie in the low-frequency spectrum. To illustrate this, we use the accelerometer as an example. Fig. 4.5a shows raw accelerometer readings of a pedestrian who encounters a sudden turn when walking inside a building. The original raw data have such a dense signal distribution over the whole recordings that it is



(a) Accelerometer raw data (b) Filtered data with $\alpha = 0.15$

Fig. 4.5: Turning event is more evident after filtering movement noises

hard to discover important event from the time domain. Therefore, we apply a low-pass filter to this sequence. In particular, we choose a moving average filter for noise reduction, which derives the moving average from the original sensor data as

$$y_i = y_{i-1} + \alpha * (x_i - y_{i-1}),$$

where the filtered sample y_i is based on its previous value y_{i-1} and the current x_i with parameter α lying between 0 to 1. As illustrated in Fig. 4.5b, by applying this filter the data are properly smoothed and the turning motion is more evident from the original noisy data.

4.2.4 Feature Extraction

Filtered exemplars are still in the form of raw sensor signals unsuitable for learning an effective model. To reduce the data volume for learning, we need to extract significant low-dimensional features from these exemplars. Features are commonly used in classification tasks to capture the properties of signal behavior. Further, since we assume the adversary has no prior domain knowledge on sensor patterns, this feature extraction and selection process should be fully automated without human intervention. In MISSILE, we adopt the FRESH procedure [16] to build an automatic significant features extractor, which remarkably reduces the effort on feature engineering. The detailed procedure is shown in Algorithm. 5, which consists of the following three phases.

4.2.4.1 Extraction of Feature Candidates

The raw time-series exemplars are first mapped into common features with a set of predefined parameters. Let us assume that there are k exemplars in the collection $E = \{E_1, E_2, ..., E_k\}$. For each exemplar, n samples are

Algorithm 5 Significant Features Extractor

```
Input:      Exemplars
            $E = \{E_1, E_2, \ldots, E_k\}$, $E_i \in \mathbf{R}^{m \times n}$
            Location labels
            $L = \{l_1, l_2, \ldots, l_k\}$
            Rank threshold $r$
Output:     Final features
            $F' = \{F'_1, F'_2, \ldots, F'_k\}$, $F'_i \in \mathbf{R}^{m \times r}$
Procedure:
 1: $F = \emptyset$, $tempPV$=$\emptyset$, $PV = \emptyset$, $Type = \emptyset$, $F' = \emptyset$
 2: for $E_i$ in $(E_1, E_2, \ldots, E_k)$ do
 3:     $F_i$ = ExtractCommonFeatures($E_i$)
 4:     Add $F_i$ into $F$
 5: for feature type $f$ in $F$ do
 6:     if $f$ is binary feature then
 7:         $tempPV$ = FisherTest($f$, $L$)
 8:     else if $f$ is real-valued feature then
 9:         $tempPV$ = MannWhitneyTest($f$, $L$)
10:     Add $tempPV$ to $PV$
11: $Type$ = RankFeature($PV, r$)
12: $F'$ = SelectFeature($F, Type$)
13: Return $F'$
```

collected from each sensor axis and a total of m sensor axes are sampled. As such, each exemplar can be written as

$$E_i = \{(s^1_{it1}, s^1_{it2}, \ldots, s^1_{itn}), \ldots, (s^m_{it1}, s^m_{it2}, \ldots, s^m_{itn})\},$$

where $t1$ is the starting timestamp of exemplar E_i.

The features F_i of exemplar E_i are extracted from various statistics on samples including maximum, minmum and root mean square ($f_i^{rms_m} = \sqrt{\frac{\sum_{t1}^{tn} |s_i^m|^2}{n}}$):

$$F_i = \{(f_i^{max_1}, f_i^{min_1}, f_i^{rms_1}, \ldots), \\ \ldots, \\ (f_i^{max_m}, f_i^{min_m}, f_i^{rms_m}, \ldots)\}.$$

4.2.4.2 Statistical Hypothesis Testing

After all the features are extracted, we need to select significant ones from them before feeding them into a classifier for learning. Statistical hypothesis test is conducted on each feature to evaluate its relevance to locations. The main idea is that, if a feature f can distinguish a particular location j_a, its conditional probability distribution on this location j_a, $P(f|j_a)$, must be significantly different from $P(f|j_b)$, the distribution on any other location j_b. Using this principle, the null hypothesis H_0^f and alternative hypothesis H_1^f to test relevance of feature f to location j_a are formulated as

$$\begin{aligned} \forall j_b \neq j_a, H_0^f &= \{P(f|j_a) = P(f|j_b)\}, \\ H_1^f &= \{P(f|j_a) \neq P(f|j_b)\}. \end{aligned}$$

A set of probability values (p-value) PV will be returned after the tests. A smaller p-value suggests stronger evidence to reject the null hypothesis H_0^f, which means the feature is relevant to location j_a against location j_b since they do not share the same conditional distribution. In MISSILE, we use two hypothesis tests, namely, Fisher's exact test for those binary features and Mann–Whitney rank test for those real-valued features.[6]

4.2.4.3 Selection

In the final step, we sum up the total p-values across all axes for each feature and rank them in ascending order. Only top-r features which have the smallest p-values are selected as the refined feature set F'.

In MISSILE, we apply FRESH [16] with over 60 categories of pre-defined features. They can be divided into two sets. Time-domain features such as mean, variance, median, and the number of peaks mainly characterize signal intensity as in time series. For example, a magnetic field sensor produces different number of peaks based on the magnetic local variation in different locations. Frequency-domain features capture the characteristics of signal pattern in terms of frequency envelope and certain frequency component after Fourier transform. Certain location such as a winding corridor may not have obvious time-domain pattern but it has unique pattern on frequency domain. Table 4.2 shows the top-12 features after the selection step using our exemplar dataset. These features constitute the inputs for location classification task in our experiment.

4.2.5 Modeling

In the core of MISSILE, we want to identify sensor patterns for different sensitive indoor locations, which is a typical classification task. There are a

[6]The Mann–Whitney rank test can examine the distribution of two real-valued random variables using statistics derived from ranking against each other. In the case of multiple locations, the test is conducted in one-vs-all style.

Table 4.2: Empirical feature extraction result on 4 selected sensors

Feature Category	**Description**	**Dimension**	**Average p-value**
VAR	Variance of each axis s in a exemplar	1	$1.18 * 10^{-5}$
LSTD	Whether the standard deviation is larger than 0.25 times of max(s)-min(s)	1	$1.56 * 10^{-3}$
MEAN	Overall average of each axis s in a exemplar	1	$4.52 * 10^{-5}$
MEDIAN	Median value of each axis s in a exemplar	1	$5.92 * 10^{-5}$
SPKT	Cross power spectral density on the second coefficient	1	$6.24 * 10^{-6}$
RRSIGMA	Ratio of values more than the distance of 2 away from mean value	1	$1.99 * 10^{-6}$
PEAKS	Number of amplitude maxima with least support of 1 and 3	2	$2.66 * 10^{-4}$
LINTREND	Linear least-squares regression value over aggregated sequence with chunk size of 50	1	$9.10 * 10^{-4}$
FFTCOE	First and third Fourier coefficients of discrete Fourier Transform	2	$3.55 * 10^{-6}$
SYM	Symmetric shape of distribution with the level of 0.1	1	$9.94 * 10^{-4}$

number of classification models suitable for this task, such as naive Bayesian, decision tree, support vector machine and neural network. All of them are capable of learning hidden pattern from data to labels. In MISSILE, we choose the non-parametric decision tree as the classifier. In particular, we use the CART (Classification And Regression Tree) classifier model [10]. This model recursively splits input attributes (i.e., features in training data) to generate a binary decision tree, where each leaf node corresponds to a class label. To split attributes, $gini$ index is employed for the impurity measurement function $H(\cdot)$:

$$H(P) = 1 - \sum_k p_k^2,$$

where p_k is the ratio of instances with label k among all the instances in node P. If $H(P) = 0$, then this node becomes a leaf node as only one label exists among all instances. We determine the order of attributes to split using $gini$ gain:

$$Gain(P) = H(P) - \sum_c \frac{|P_c|}{|P|} H(P_c),$$

where P_c represents child node c of node P, and $|\cdot|$ means the number of instances. A higher gain value indicates a better choice to split this node.

4.2.6 Calibration for Anomalies

In the above classification, we trust the training set with their labels and input features. However, in reality there are anomalies in the training set. First, exemplars from the automatic collection may be labeled with incorrect location due to BLE signal delay or signal penetration from the floor or wall. Second, malfunctioned mobile sensors may produce low-quality data, which significantly contaminates the training set. To prevent the above anomalies from degrading the classifying accuracy, we adopt two orthogonal machine learning techniques, namely Ensemble Learning (e.g., Random Forest [9]) and Isolation Forest [84] to identify these anomalies.

4.2.6.1 Random Forest

Ensemble learning uses multiple learning algorithms with bootstrapping technique to achieve better classification accuracy than could be achieved from any of the constituent learning algorithms alone. Recent side-channel attack research [72] [36] suggests that an ensemble version of the decision tree, namely the random forest, is suitable to conduct learning tasks on a noisy dataset with distinguishing patterns. Random forest generates a multitude of decision trees, and trains each with random subset data of the given features. When classifying input features, the data pass through every tree in the forest and the final prediction is decided by the most predicted label of these trees. The various trees trained with different subset data provide significant variability for a prediction model, thus reducing the overfitting issue. However, an ensemble classifier is at the cost of consuming more CPU time for training and classification.

4.2.6.2 Isolation Forest

Isolation forest is a robust and efficient anomaly detection algorithm with linear time complexity. It can be used before the training phase to filter anomalies. The core idea of isolation forest is that anomalies are sensitive to isolation when separating attributes. In other words, an anomaly is usually far away from the dense distribution inside the class cluster, so it is singled out at the early stage of isolation. Essentially an isolation tree (i-Tree) is a full binary tree with random attribute split. It first randomly picks a feature from an input feature set and selects a random splitting point between the minimum and the maximum value of this attribute. All the instances are then separated into two partitions based on this splitting point, one assigned to the left child node and the other to the right child node. The isolation is then recursively conducted until all instances are isolated in the leaf nodes.

Isolation forest constitutes multiple i-Trees obtained by isolating different subsets of the original set of n instances. An anomaly score is then given to each instance x as follows

$$Score(x,n) = 2^{-\frac{E(h(x))}{\mathcal{A}(n)}},$$

where the nominator is the expectation of the path length $h(x)$ (i.e., number of edges from the root node to the x instance) of all i-Trees inside the forest. And the denominator $\mathcal{A}(n)$ is the average path length of an i-Tree given n samples (i.e., average number of edges from the root node to any external leaf node) to normalize $E(h(x))$, which can be derived by

$$\mathcal{A}(n) = 2H(n-1) - \frac{2(n-1)}{n},$$

where $H(n-1)$ is the $(n-1)$-th harmonic number.

Once the scores are ready, we filter those instances whose anomaly scores are higher than our designated threshold. The final score is in $(0,1]$ since the fractional component in $Score(\cdot)$ is greater than zero and it is bounded by an exponential function. An exemplar is considered as an anomaly when its score is close to 1 (i.e., $E(h(x))$ is much smaller than average path length) and a normal one when it is close to 0 (i.e., $E(h(x))$ is much greater than average path length).

4.2.7 Attacking Stage

Unlike training stage, the attacking stage, i.e., location eavesdropping, is operated on the victim's device. As such, the key challenge in this stage is to operate in a stealthy manner, i.e., using as low footprint of CPU, memory, bandwidth and power as possible. Regarding low CPU and memory footprint, we employ *sliding window* [45] to process the sensory data stream by limiting the extent of data to a sequence of most recent samples. It is usually defined by tuple $\{win, str\}$ where win is the range of windowing and str is the stride when sliding. When it is applied to the attacking stage, the sensor data are sliced by the sliding window to form a specific length of exemplar and fed to the embedded classifier. Once a sensitive location is inferred, the spyware can take various actions such as notifying its command-and-control center or starting audio recording (if corresponding permission has been granted).

Regarding low power footprint, we propose two optional techniques to reduce the activity of the spyware. The main idea is to invoke the location eavesdropping only when the victim is walking and is not far away from a sensitive location.

Opportunistic Wi-Fi Activation: Nowadays many buildings or common areas are covered by a large public Wi-Fi. By scanning the available SSIDs, the spyware or repackage application can activate eavesdropping only when a victim device "sees" a specific SSID, which means it is close to the premises concerned. Note that scanning nearby SSIDs may require Internet-related permission in the recent release of operating system[7], but most users tend to grant it because it is the most common permission.

Motion Activation: To reduce unnecessary eavesdropping activity when victim is non-moving (standing or sitting still), the spyware can start monitoring only after a motion is detected on the victim through endpointing. Endpointing is a common technique used in speech recognition system to determine the start and end of a user speech and to separate speech region and non-speech region [80]. We can apply endpointing in motion sensor data as the energy in movement region, i.e., the sum of squared sample values,

[7]Android does not restrict on scanning SSID until Oreo (8.0). Even in Oreo, SSID scanning is still allowed if an app has any of the three permissions (*CHANGE_WIFI_STATE, ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION*).

is typically much higher than the energy in non-movement region. We can allow sensors to deep sleep and wake up intermittently to see if the current average window of energy exceeds a pre-defined threshold. If it does not exceed, all sensors continue to sleep until the next wake-up cycle.

4.3 Performance Evaluation

To evaluate the real-life performance of the MISSILE system, we conduct experiments on sensitive locations in a university campus, including students' laboratory, professor's office, common room, washrooms, ATM station, and canteen entrances. The disclosure of these locations can lead to significant privacy breach where, for example, the frequency of accessing washrooms and ATM can indicate personal health and financial status. In practice, entrances, exits and corridors connecting different zones are good targets of sensitive locations as they can be used to outline a victim's daily activity. Such knowledge can further lead to social engineering attacks. As for the selection of locations, we first identify sensitive indoor areas that imply strong semantics of personal activities and may arouse interests of attackers. Then for each chosen location we represent it (and its neighborhood) by a combination of visual characteristics (e.g., door, turn, corridors) as listed in Table. 4.3. In our experiment, we choose 15 representative sensitive locations that exhibit different combinations of visual characteristics, which constitute the unique patterns when victims pass by. Fig. 4.6 shows photo snapshots of four sample locations whereas Fig. 4.7a and Fig. 4.7b plot them in their corresponding floor plans.

The sensory data are collected by 10 individuals with mixed genders and body figures. They carry the test devices with random placement (left/right/front pockets) for their daily use over a period of 90 days. As for location labels, we take advantage of existing BLE beacons deployed by other services (e.g., teaching facilitation) to label sensor data and each location has one beacon as reference point. To preclude the impact of the way we split training and test datasets, all experiments are conducted 10 times using random splits and the averaged results are reported. Specifically, among all 2580 exemplars of sensitive locations, 6 individuals' exemplars (around 1548 exemplars) are used for training while the other 4 individuals' (around 1032 exemplars) are used for testing. As for non-sensitive locations, we randomly extract 350 exemplars for training into a "non-sensitive location" class, which is close to the number of exemplars of the most popular sensitive location.

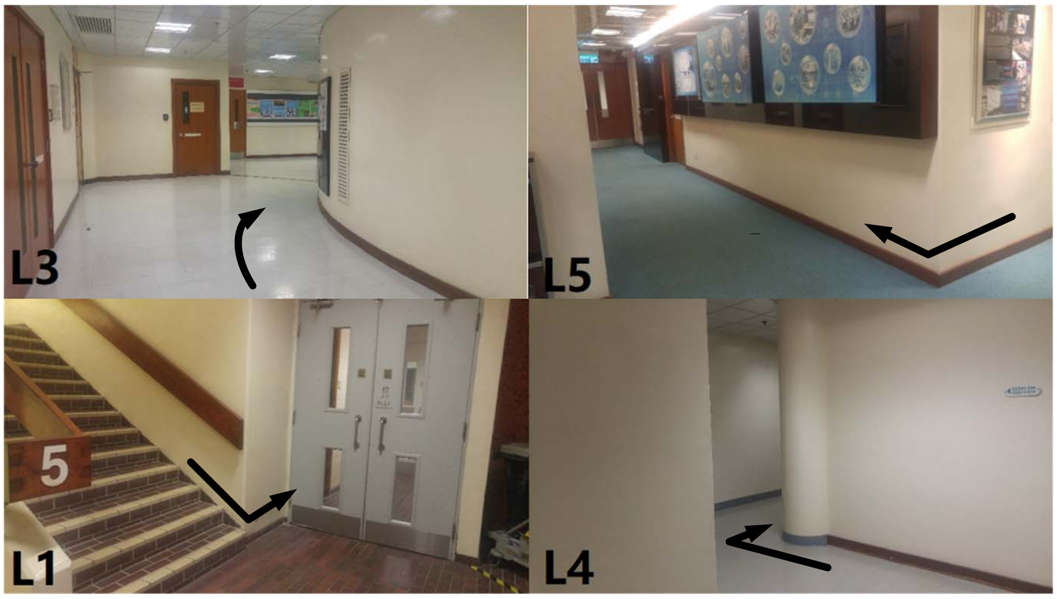
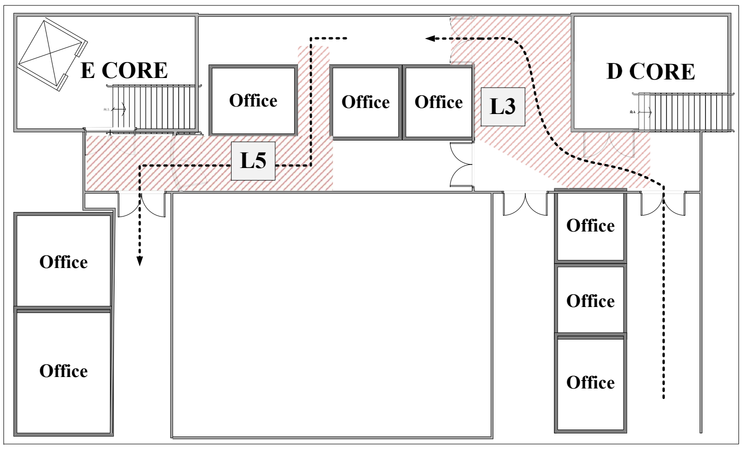
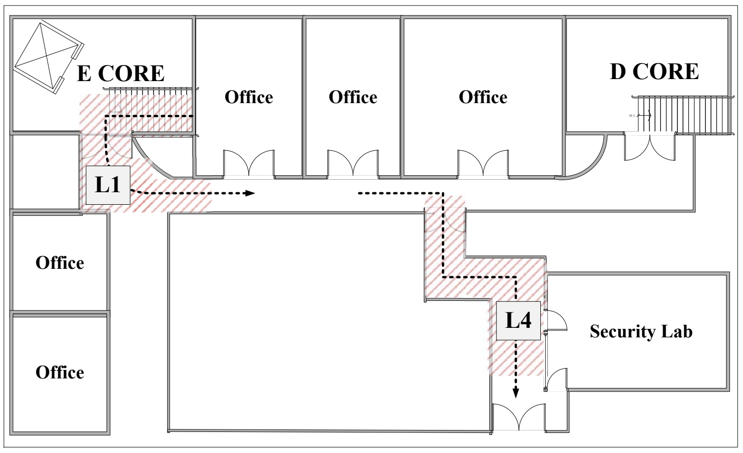


Fig. 4.6: Indoor sensitive location examples



(a) Floor plan for $L3$ (falculty office) and $L5$ (inventory office)



(b) Floor plan for $L1$ (research laboratory) and $L4$ (teaching laboratory)

Fig. 4.7: Experiment floor plan with example trajectory (shadow area illustrates sensitive area).

Table 4.3: Visual Characteristics of Sensitive Locations

Vital Characteristics	Location ID														
	$L1$	$L2$	$L3$	$L4$	$L5$	$L6$	$L7$	$L8$	$L9$	$L10$	$L11$	$L12$	$L13$	$L14$	$L15$
Single Fire Stop Door		✓	✓	✓	✓						✓		✓	✓	✓
Double Fire Stop Door	✓												✓		✓
Corridor Quarter Turn		✓		✓	✓			✓				✓			
Winding Corridor	✓		✓			✓			✓		✓				
Straight Corridor						✓								✓	
Elevator	✓						✓	✓							
Stairways	✓	✓					✓			✓					

For testing, we extract another 1400 exemplars with non-sensitive locations. This ratio, $1032 : 1400$, approximates the statistics of ratios of sensitive to non-sensitive locations of all exemplars in our experiment. Other system parameters are listed in Table 4.4.

Our comparative study includes: (1) the performance of different normalization approaches (i.e., rotation-matrix-based versus Euclidean-norm-based normalization), (2) the impact of ensemble classifiers (i.e., decision tree versus random forest), (3) the impact of isolation forest, (4) the impact of training data size, (5) the impact of sensor types, (6) the impact of different location characteristics, (7) the impact of system parameters (e.g. exemplar length, tree numbers in random forest and feature setting), and (8) the power consumption on popular devices. To evaluate the effectiveness of MISSILE attack, we categorize all classification results of location label i into 4 cases in one-vs-all style: true positive (TP_i, recognizing a location i correctly), true negative (TN_i, ignoring a location i correctly), false positive (FP_i, recognizing a location i incorrectly), and false negative (FN_i, ignoring a location i incorrectly). Based on these cases, we define $precision$ and $recall$ for each location label i as follows

$$precision_i = \frac{|TP_i|}{|TP_i| + |FP_i|},$$

$$recall_i = \frac{|TP_i|}{|TP_i| + |FN_i|}.$$

The $precision_i$ essentially tells how well the system can distinguish location i from other locations while $recall_i$ shows how well the system can detect a particular location label. As these two metrics are sometimes contradicting to each other, we also employ the F1-score [14] as an overall metric for each location label i, which is

$$F1_i = \frac{2 \cdot precision_i \cdot recall_i}{recall_i + precision_i}.$$

The overall F1-score is the weighted average F1-score of all location labels, based on the number of true instances of each location in the testing data label set L as follows

$$F1_{overall} = \sum_i \frac{|L_i|}{|L|} F1_i.$$

Table 4.4: System Parameters

Parameter	Value
Exemplar Length	15s
Sampling Frequency	50Hz
Noise Filter α	0.15
Step Threshold	750
RSSI Threshold	-85dB
Features	12 features per sensor axis
Anomaly Threshold	0.8
Models	Decision Tree, Random Forest
Devices	(Android Version, RAM, CPU) LG G3 (5.0, 3GB, 2.5GHz), Redmi Note4X (6.0, 4GB, 2.0GHz), Google Pixel (7.1, 4GB, 2.15GHz), HTC U Ultra (8.0, 4GB, 2.15GHz), Samsung Galaxy S8 (8.0, 4GB, 2.35GHz)
Selected Sensors	Gyroscope (3-axis), Magnetic Field Sensor (3-axis), Linear Acceleration Sensor (3-axis), Accelerometer (3-axis)

Table 4.5: Overall Performance of Missile System

Classifier	Training Time (s)	Inference Time (ms)	F1-Score	F1-Score (after Isolation Forest)
DTEN	0.6 - 0.8	0.8 - 1.2	35.14%	42.35%
DTRM	1.6 - 2.0	1.5 - 2.0	49.27%	53.13%
RFEN	9.0 - 10.0	40.0 - 60.0	59.62%	63.14%
RFRM	15.0 - 16.0	60.0 - 80.0	70.81%	73.79%

4.3.1 Overall Performance of Missile System

Table. 4.5 shows the performance comparison between decision tree (DT) and random forest (RF), with Euclidean-norm-based (EN) and rotation-matrix-based normalization (RM), namely, DTEN, DTRM, RFEN, RFRM. We observe that all classifiers significantly outperform random guess (one out of 16 choices, 6.25%), which justifies the feasibility of MISSILE. Further, random forest, an ensemble classifier, can achieve an even higher F1-score of 62%. On the other hand, rotation matrix normalization always outperforms Euclidean norm by at least 10%, because it preserves useful information for classification. Anomaly detection by isolation tree has shown moderate improvement of F1-score for all classifiers, among which the classifier with

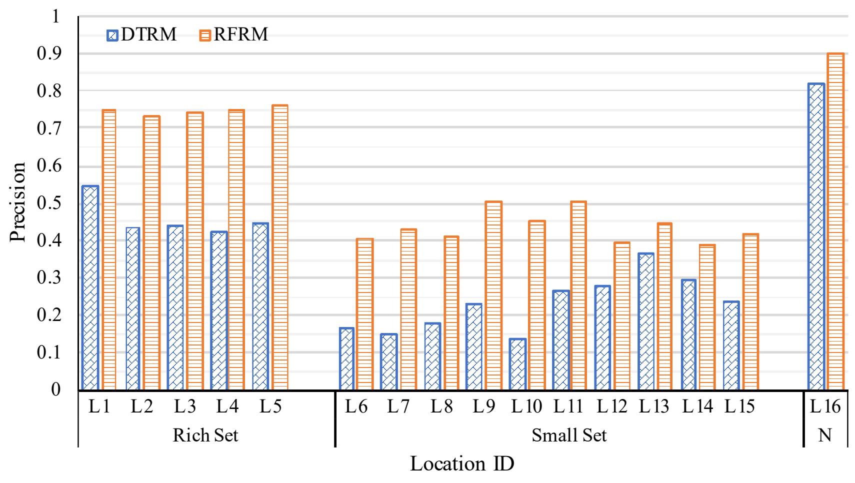


Fig. 4.8: Impact of training data size by decision tree and random forest

decision tree and Euclidean norm witnesses over 6% improvement. This shows both the ensemble method and rotation matrix normalization are more robust against anomalies. For the rest of experiments, we will mainly report DTRM and RFRM results after isolation tree. In terms of CPU time, classifiers using ensemble method cost significantly 10 times more CPU resources to train the model and 50 times more to make a prediction. Rotation matrix normalization also noticeably increases the training overhead but has less influence for prediction. It suggests that spyware can switch among different classifiers to balance battery condition and desired utility.

4.3.2 Impact of Training Data Size

Fig. 4.8 illustrates the precision for individual sensitive locations. We categorize them into locations with small training data (≤ 50 exemplars) and locations with rich training data (≥ 150 exemplars). We observe that locations in the former category ($L6$ to $L15$) have a higher probability to be misclassified. This effect is more eminent for the decision tree than for the random forest, as the former is a single classifier method and thus more vulnerable to noises and outliers. An ensemble method such as the random forest tends to alleviate the impact of noises and outliers by splitting data into subsets with crossover items, so that they cannot easily dominate the training process. Note that $L16$ (grouped as N) is a location label for all non-sensitive locations. It achieves around 90% accuracy in $RFRM$, which indicates that the system is able to identify most of non-sensitive locations.

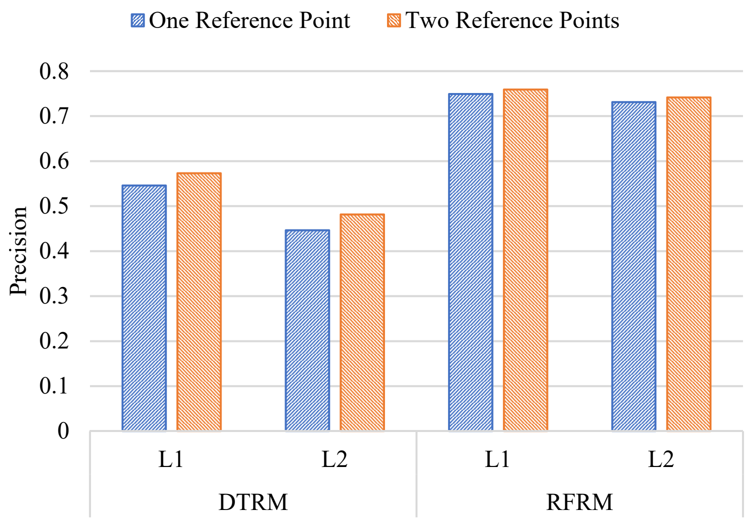


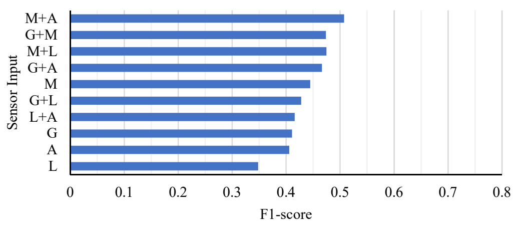
Fig. 4.9: System precision with one and two reference points calibration

4.3.3 Impact of BLE Reference Points

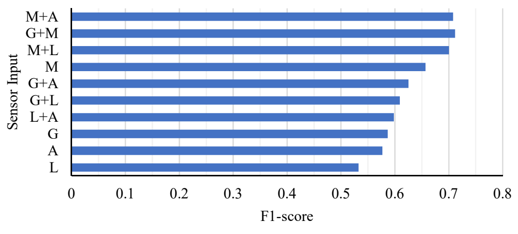
To evaluate the impact of the number of BLE reference points on the annotation of starting timestamps, we collect additional training data from location L1 and L2 using two beacons each and re-train our system. Fig. 4.9 plots the difference of system precision using one and two reference points. Overall, the system with two reference points always performs better by 3%-4% for DTRM and 1% for RFRM. The small gain might be attributed to the long exemplar length, which is already long enough to include enough sensor patterns to distinguish a sensitive location (more details are discussed in Section 4.3.6. Since one reference point already leads to satisfactory performance, throughout the experiment we use one reference point for each location and calibrate the starting timestamps of exemplars with the mean of deviation distribution.

4.3.4 Impact of Sensor Type

To investigate the contributions of different sensor types in the MISSILE system, we measure the F1-scores using single or a pair of sensors in Fig. 4.10. We observe that in both DTRM and RFRM, the top F1-score rankings are similar, which means some sensor or sensor combinations are consistently better than the others regardless of the classifiers. In particular, the magnetic field sensor plays a major role, with its F1-score reaching over 60% (alone) and around 70% (pairwise). This indicates that the magnetic distribution caused by geolocation and indoor structure material can constitute a unique signature for inferring sensitive locations. By combining another motion sensor, such a sensor pair can approximate the result of using all four sensors.



(a) Sensor evaluation with DTRM



(b) Sensor evaluation with RFRM

Fig. 4.10: Performance results in descending order for different sensors: Gyroscope (G), Magnetic Field Sensor (M), Linear Acceleration Sensor (L), Accelerometer (A)

Other three sensors leverage user behavior information and obtain similar results (over 50%).

4.3.5 Impact of Location Characteristics

Regardless of the methods used for classification, we observe that the F1-scores in some locations are consistently better than those in the others. For example, locations $L1$ and $L2$ have both high precision (75% and 73%) and high recall (90% and 86% in Fig. 4.11). From Table 4.3, we learn that $L1$ and $L2$ have 4 and 3 characteristics, respectively, while all other locations have 2 or even fewer. Furthermore, some characteristic has more significant impact than the others. For example, the top-ranked recall locations — $L1$-$L5$, $L13$, $L14$, $L11$, and $L15$ — all share a common characteristic: a fire stop door. Such high recall implies a high tendency to identify locations with door opening event correctly.

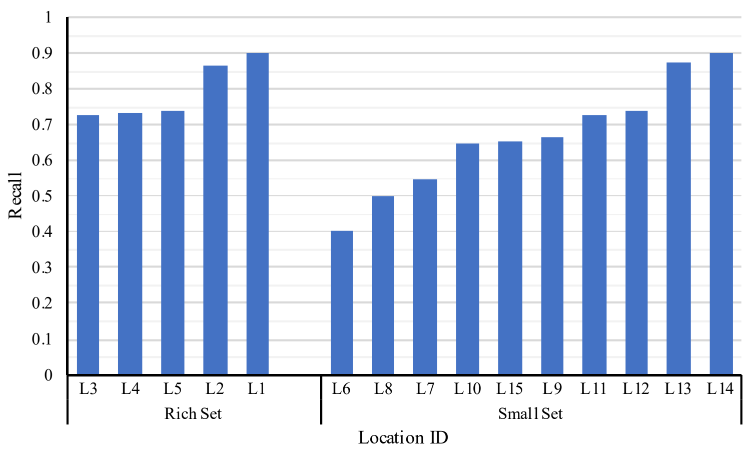


Fig. 4.11: Recall of RFRM in ascending order for individual location

4.3.6 Impact of System Parameters

4.3.6.1 Impact of Exemplar Length

In previous experiments, we set 15 seconds as the standard time length of an exemplar in the attacking stage. This value is set to generate a sufficiently long signal pattern that captures necessary characteristics of any sensitive location. In this subsection, we vary this length from 3 to 30 seconds and plot the F1-score in Fig. 4.12a under both decision tree and random forest methods.

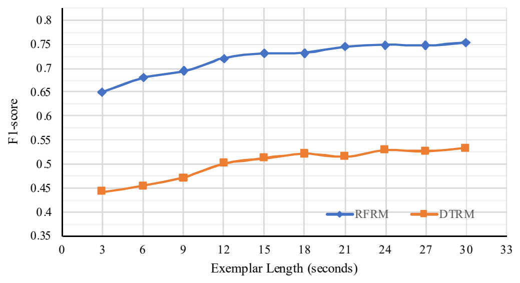
For both methods, we observe a steady increase as the exemplar length increases, which coincides with our reasoning above that a longer exemplar may capture more characteristics of a sensitive location. However, the F1-score starts to saturate after 12 seconds especially for classifier $RFRM$, which indicates that over-extending this length does not significantly help to further improve the classification results as the chance of a sensitive location being covered by two consecutive exemplars is slim.

4.3.6.2 Impact of Random Forest Setting

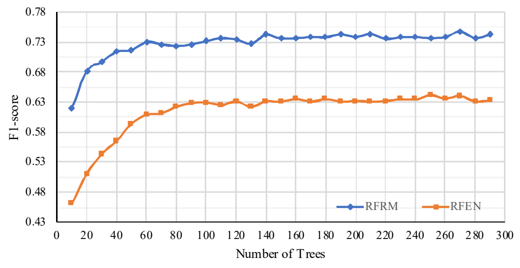
We vary the number of decision trees for the ensemble method (i.e., the random forest) and plot the F1-score for both Euclidean norm and rotation matrix normalization in Fig. 4.12b. We observe that both methods reach a saturation point after 60-100, which means the random forest is robust under this parameter.

4.3.6.3 Impact of Feature Setting

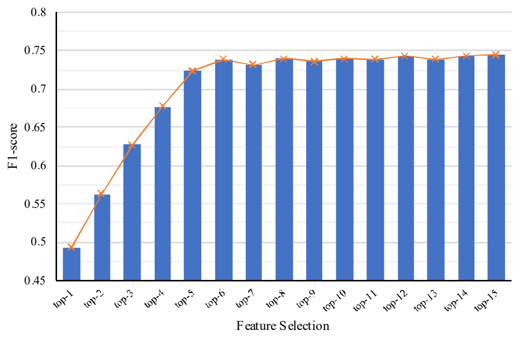
Top-12 feature selection is adopted during the automatic feature extraction in all the previous experiments. To examine the impact of this setting, we measure and plot the F1-score change of classifier $RFRM$ with feature sets generated under different top-r settings in Fig. 4.12c. We observe that top-1 feature in classifier $RFRM$ can reach an F1-score of 48% alone. The performance of classifier grows steadily until this setting reaches top-6 and F1-score saturates at around 73%. This indicates that a minimum of top-6 feature setting is required for classifier $RFRM$ to achieve its best performance. Since the ranking is decided by p-values shown in Table. 4.2, it is obvious that features containing the unique information of sensitive location are highly associated with low p-values.



(a) System performance under different exemplar lengths



(b) System performance under different number of trees in random forest



(c) System performance of RFRM under different top-r settings

Fig. 4.12: Impact of system parameters over performance

Table 4.6: Spyware Power Consumption on Monitoring Sensors (50Hz)

Model	Capacity	Spyware-On	Idle	Usage
HTC U Ultra	3000mAh	1.323%	0.611%	0.712%
Samsung Galaxy S8	3000mAh	1.529%	0.801%	0.728%
Google Pixel	2770mAh	1.317%	0.507%	0.810%

4.3.7 Power Consumption

The spyware installed by MISSILE continually samples multiple mobile sensors. To reduce power consumption, we implement both opportunistic WiFi activation and motion activation as in Chapter 4.2. To further evaluate the power impact of continuously accessing sensors, we activate the spyware in the background to sample sensors with the screen off and measure the power usage per hour of various smartphones. The result is presented in Table. 4.6, which shows a moderate consumption of around 0.7% - 0.8% extra battery per hour.

4.4 Extension and Countermeasure

In this section, we will discuss the potential extension of MISSILE and countermeasures.

In the experiment, the sensitive locations data are collected by attackers manually, which limits the scalability of this attack. To acquire a large-scale and more diversified dataset, this process can be enhanced by automation or crowdsensing. The former, such as IndoorAtlas [37], can provide efficient sensory measurement of indoor location. The latter delegates the task of sensing and labeling location data to a crowdsourcing platform.

Currently, the MISSILE system only considers stateless recognition, which does not take the relationship between sensitive locations into consideration. Inspired by dead reckoning for indoor positioning [44], we can improve the location inference performance by extracting detailed context such as walking distance, turning angle and pushing motion.

As for countermeasures for indoor sensitive location inference attack of MISSILE, we propose two methods as below.

Access Control: Permission mechanism is the first line of defense. We suggest that no request from mobile application for statistics or raw sensory information should bypass the permission mechanism. In addition, since high-resolution sensor data can be exploited by attackers who take advantage of subtle change [62], we suggest replacing them with feature-level data access, which also significantly reduces computational cost.

Data Manipulation: Noise injection is an alternative countermeasure. Software level noise injection has already been applied to GPS data in the geo-social network. With the same rationale, noise can be injected into sensor readings to avoid highly accurate location inference. If the operating system cannot be trusted to perform this injection, we recommend employing hardware noise injection, for example, enabling the vibrator of a mobile device.

4.5 Summary

In this chapter, we investigate a side-channel attack that can eavesdrop user's sensitive locations using unprivileged sensory information. This attack is modeled as a classification problem of various sensory data collected from different locations. The classifier is built from supervised learning of training data prepared by automatic labeling mechanism, effective processing and optimal feature extraction. Real experiments are conducted on 15 indoor locations inside a university campus. The classifier using modeling with anomaly calibration can reach around 73% F1-score, which is significantly higher than random guess. As for future work, we plan to implement the improved version using multiple reference points for labeling, stateful routes and other side-channels (e.g., JavaScript in mobile browser). We also plan to investigate countermeasures against such attack, evaluate and compare them on various metrics, such as time complexity, accuracy and utility.

5 Collecting Data with Privacy Protection

In this chapter, we focus on the attack surface of individual records and how to prevent privacy leakage during collection. In most of the machine learning applications, the machine learning model is refined by continually feeding in new user data (as features) and their feedback (as labels) from their mobile devices. However, these data, such as type history, web access logs, and frequently visited locations, are often sensitive and private information. Despite of strict legislation on personal data protection and the efforts made by most service providers, hosting personal data in a centralized location can still be highly risky due to security breach, internal theft or corporate dishonesty. A famous incident is the leakage of celebrity photos from iCloud in 2014. Unfortunately, centralized sanitation (e.g., generalization) and encryption schemes are shown vulnerable to various attacks, such as deanonymizing Netflix challenge dataset with IMDb data [67].

More recently, two distributed data analytical tools are proposed to protect privacy, namely, local differential privacy [21] and federated machine learning [59]. Both tools avoid direct access of personal data while still retaining high utility, e.g., high accuracy on statistics estimation or the trained model. Their mechanisms are summarized as follows:

1. Local differential privacy (LDP): Each user perturbs her data locally before sending them to an untrusted service provider for data collection and analytics. LDP achieves plausible deniability of each individual under a measurable and rigorous mechanism. LDP is heavily investigated in the literature of privacy-preserving statistics collection.
2. Federated machine learning (FML): It trains a globally shared model over a large number of distributed clients using an efficient control protocol with the central server. Only model parameter updates calculated on local data are submitted to the server, who aggregates them to improve the shared global model. This approach not only protects users' local data but also leverages on the computing resources on mobile devices.

Although both tools avoid direct access, their methodologies are essentially different. LDP is a theoretical privacy notation that can be achieved by different algorithms, while FL is a generic distributed learning framework without theoretical provable privacy. To conduct a comparative study of both tools, we deploy them to solve a common set of classification problems in mobile scenarios. This allows us to gain important insights of their performance in terms of classification performance, privacy loss, CPU/power consumption, and bandwidth consumption. In particular, to unify the privacy model of both solutions, we design a privacy loss metric through a general sample inference attack. To summarize, our primary contributions are as follows:

- We implement two competing solutions that learn from user data without submitting the original user data to the server and extensively discuss the unification of two solutions.
- We design a unified privacy loss metric for both solutions through a general sample inference attack.
- We conduct extensive experiments to compare both solutions in a set of machine learning problems in mobile scenarios.

The rest of the work is organized as follows. In Chapter 5.1, we introduce the fundamental principles of LDP and federated machine learning, and point out their problems. Chapter 5.2 presents the methodology of our comparative study of the two techniques. The experimental results shown in Chapter 5.3 compare their performance for given learning tasks with respect to various model and dataset parameters. We discuss other challenges of FML in Chapter 5.4. And finally, the findings of the study are summarized in Chapter 5.5.

5.1 Background

5.1.1 Local Differential Privacy

LDP [21] extends the notion of differential privacy by perturbing local data with noise determined by a predefined parameter. In a nutshell, a perturbation algorithm $\mathcal{A}$ probabilistically modifies a local raw value ν_i to another value in the same domain of possible outputs κ. The modified value is then submitted to the server. A learning task on the statistical features (e.g., frequency and mean) of such data retains certain accuracy after the server collects all perturbed values. Meanwhile, each individual can have plausible privacy guarantee bounded on a privacy budget of ϵ.

Formally, the perturbation algorithm suffices ϵ-LDP principle if and only if for any two individuals' inputs ν_i and ν_j, we have

$$Pr[\mathcal{A}(\nu_i) = s] \leq e^{\epsilon} \cdot Pr[\mathcal{A}(\nu_j) = s],$$

where $s \in \kappa$. Obviously, perturbed data is closer to the original data with a larger privacy budget ϵ and user population. Since the noises are applied to the data set directly, this strategy may have a strong impact on model performance when the budget is low.

5.1.2 Federated Machine Learning

In a task of federated machine learning, each mobile device initializes its own training using the shared model downloaded from the server and builds a new model using its local data. The updated model parameters are then be returned to the server, averaged with other peer devices and merged as the new shared model. This process is repeated multiple rounds to satisfy a learning objective until the desired set of model parameters are obtained.

Formally, a typical supervised machine learning objective function can be expressed as

$$\arg\min_{W} \frac{1}{N} \sum_{j \in J} \mathcal{L}(f(x_j, W), y_j),$$

where a learning algorithm is stated as f and its corresponding parameters W are estimated from the dataset J with a total sample size of N by minimizing the loss $\mathcal{L}$ between predictions on all input x_i and true label y_i in the training set.

In federated machine learning, data are assumed to be distributed over a set of M mobile devices and each of them can be considered as a partition P with $n = |P|$ training samples. The objective in this setting evolves to minimize the aggregated loss:

$$g(W) = \sum_{m \in M} \frac{n_m}{N} F(P_m, W), \tag{5.1}$$

where F is the local loss defined by

$$F(P_m, W) = \frac{1}{n_m} \sum_{k \in P_m} \mathcal{L}(f(x_k, W), y_k). \tag{5.2}$$

To train such an objective in Eqn. 5.1, a straightforward gradient descent algorithm can be applied to estimate model parameters using the iterative rule below:

$$W_{t+1} \leftarrow W_t - \eta \nabla g(W),$$

which is a full-batch gradient descent using all client data to generate an update in round t. However, this is not practical since it takes a long time for each iteration and even multiple times longer under the case of potentially high latency and limited bandwidth of the mobile network. To improve communication efficiency, federated machine learning commonly increases individual client computation by asking each mobile device to iterate over local data several times with stochastic gradient descent before submitting the parameter updates to the server for averaging [59].

5.2 Methodology

5.2.1 Problem Statement

We aim to tackle a machine learning problem in a distributed data setting where companies such as Google and Apple would like to improve their AI service accuracy, such as word auto-complete suggestion, through the data (e.g., keyboard input) from millions of distributed data points. To minimize the risk of privacy leakage, these companies adopt either of the two strategies: local differential privacy to allow users to perturb data before submitting to them or federated learning to train the machine learning model locally and only update the model parameters to them. Table. 5.1 summarizes the main characteristics of both strategies. A typical data record for classification task is in the form of $\{X_1, X_2, ..., X_l\}$ where X_i $(i < l)$ are feature dimensions and the last one X_l is the classification label of this record.

5.2.2 Strategy *LDP*: Submit Perturbed Data with ϵ-LDP

5.2.2.1 Client Side:

To perturb each user's data while satisfying ϵ-LDP, a sanitized mechanism is introduced which covers sensitive information with a certain amount of noises. For categorical attributes, each of the attributes has k_i $(1 \leq i \leq l)$ candidate values across all samples. For any dimension X_i, the perturbed

Table 5.1: Comparison of LDP and FML

	Local Differential Privacy	**Federated Machine Learning**
Target	Data Collection	Distributed Learning
Computation	Mobile Perturbation, Server Training	Mobile Training, Server Aggregation
Application	Shared Model	Personal/Shared Model
Privacy Preservation	Adaptive Privacy Budget	Model Updates Only
Communication	One-time Submission	Multiple Interactions
Frequent Data Type	Structured Data	Text, Image, Audio

output can be X'_i by using a staircase mechanism proposed by [43], namely k-RR:

$$P(X'_i|X_i) = \frac{1}{k_i - 1 + e^\epsilon}\begin{cases} e^\epsilon & \text{if } X'_i = X_i \\ 1 & \text{if } X'_i \neq X_i \end{cases},$$

where there will be a probability of $\frac{e^\epsilon}{k-1+e^\epsilon}$ to output the real value, and $\frac{1}{k-1+e^\epsilon}$ to output one of the remaining $k-1$ candidate values.

As for numeric attributes normalized in [-1,1], a piecewise mechanism by [90] can be applied as follows:

$$\begin{aligned} P(X'_i|X_i) &= \frac{1}{e^{\epsilon/2}+1}\begin{cases} e^{\epsilon/2} & \text{if } X'_i \in [L_i, R_i] \\ 1 & \text{if } X'_i \in [-\delta, L_i) \cup (R_i, \delta] \end{cases}, \\ \delta &= \frac{\exp(\epsilon/2)+1}{\exp(\epsilon/2)-1}, \\ L_i &= \frac{\delta+1}{2} \cdot X_i - \frac{\delta-1}{2}, \\ R_i &= L_i + \delta - 1. \end{aligned}$$

where there will be a probability of $\frac{e^{\epsilon/2}}{e^{\epsilon/2}+1}$ to output a value sampled in $[L_i, R_i]$, and $\frac{1}{e^{\epsilon/2}+1}$ to output one in $[-\delta, L_i) \cup (R_i, \delta]$. After perturbation, the sanitized data $\{X'_1, X'_2, ..., X'_l\}$ will be submitted to the server when a high-speed network is available such as Wi-Fi.

5.2.2.2 Server Side:

The server receives a set of perturbed data from clients and concatenates them into one large dataset. Different from the statistics collection which usually has a calibration, the sanitized data won't have such a post-processing step since we aim at generating a perturbed dataset. All data points will be checked for any invalid or erroneous values produced by the client side. Features can be further extracted and put into learning pipeline to train a new model. To make use of the current model, its parameters will be used to initialize the new model.

5.2.3 Strategy *FML*: Train Locally with Federated Machine Learning

5.2.3.1 Client Side:

The client receives an instruction for model update task with a set of training parameters like local batch size and the number of training passes. The current service model with weights W will be downloaded into this device. Local data will be formulated into a proper input form and put into the training pipeline. In the current round t, the client m may iterate through the local data E passes with learning rate η before uploading results using the following gradient descent,

$$W_{m_t} \leftarrow W_{m_t} - \eta \nabla F(P_m, W),$$

where P_m is local data $\{X_1, X_2, ..., X_l\}_m$ used in one iteration but this can be controlled by the server to avoid using the whole local dataset in one batch.

5.2.3.2 Server Side:

The server sends out an invitation to a fraction C of current online devices M at each round of training and starts sending service model to $C \cdot M$ devices after confirmation. Updates received from the selected clients will be merged, which is equivalent to:

$$W_{t+1} \leftarrow \sum \frac{n_m}{N} W_{m_t}.$$

This server-client interaction will be repeated for multiple times until the changes of parameters meet the pre-defined threshold.

5.3 Evaluation

5.3.1 Setup

We evaluated three public datasets in this comparative study.

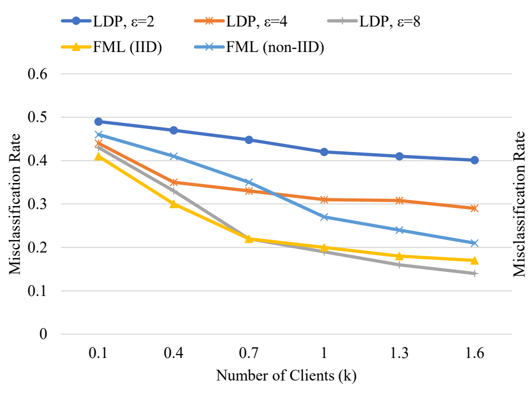
- *NYC Taxi* [85]: This dataset contains $1.4m$ samples of 2016 yellow taxi trips in New York City. Based on the 8 attributes (e.g., number of customers, starting location of a trip), a model is trained to predict the duration of each trip.
- *BR2000* [38]: This dataset has $38k$ samples of census data collected in 2000 Brazil demographic census. Based on the 13 attributes (e.g., household, disability), a model is trained to predict a person's monthly income.
- *Adult* [22]: This dataset consists of $45k$ samples of census data from UCI Machine Learning Repository. 14 attributes (e.g., education level, occupation) are provided to determine whether a person earns over $50k$ a year.

All datasets contain categorical and continuous attributes. To ensure the data are applicable to *LDP*, we perturbed the numberic attributes and categorical attributes using corresponding mechanisms. Missing values and outliers were removed. For both tasks, the machine learning model was a neural network with 2 hidden layers containing 30 units, followed by *relu* activation function. The output layer was a *softmax* activation to produce classification results. Both strategies were given an initial model trained by 10% of data. The remaining 70% of data were distributed to clients for local training and 20% were used for testing. Similarly, *LDP* only perturbed the 70% of the data and use the original 20% for testing.

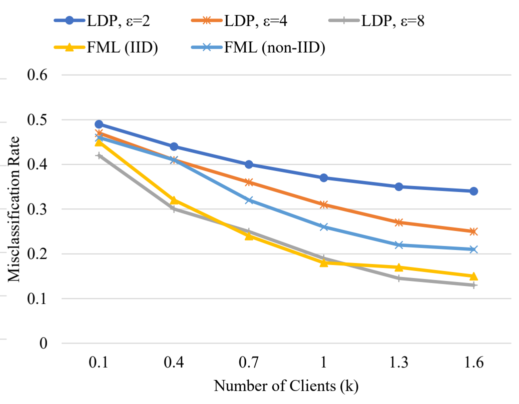
All experiments are implemented with Python 3.6 on a desktop computer running Windows 10 with Intel Core i7-7700 3.6GHz CPU and 32G DDR4 RAM. Federated learning is simulated with TensorFlow r1.13. As the experiments require thousands of mobile devices to participate, which we do not own, we use multiple server machines and multithreading to simulate these devices. For privacy budget in *LDP*, we demonstrate the results of ϵ set to 2, 4 and 8, which are common budgets adopted by industries [5]. For LDP, the central model is trained with 500 iterations for maximum 100 epochs using a learning rate of 0.1. As for FML, by default, we pick 20% of clients in each round for maximum 200 rounds and iterate 20 local passes with learning rate 0.1 in each device before uploading the updates.

5.3.2 Classification Performance

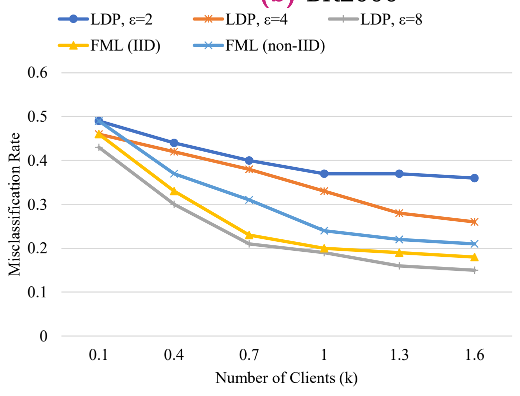
To explore the performance of two strategies, we evaluated the misclassification rate with respect to the number of clients. The rate was reported when



(a) NYC Taxi



(b) BR2000



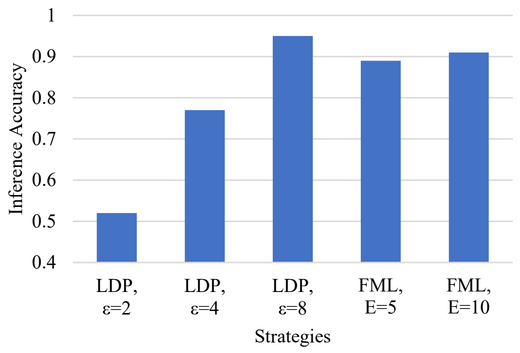
(c) Adult

Fig. 5.1: Misclassification rate of different strategies

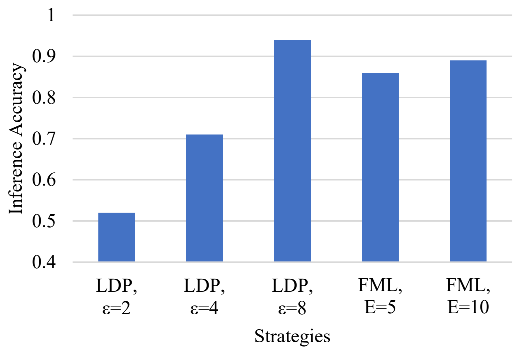
it converged during training or exceeded the maximum number of server epochs in *LDP* (resp. maximum communication rounds in *FML*).

As shown in Fig. 5.1, both strategies reduce misclassification rate with the change of client numbers from $0.1k$ to $1.6k$. The rate of *LDP* does not change much for the budget of 2 until it reaches around 1300 clients where the rate achieves the optimal 34% in *BR2000* dataset. The case with budget of 4 converges slightly quicker to a misclassification rate of 27% while the budget of 8 reaches the optimal performance of 15% in *Adult* dataset and eventually outperforms *FML* in most datasets. This is consistent with the perturbation mechanism where more relaxed privacy guarantee, i.e., greater budget, leads to lighter noises. Most of the misclassification rates saturate after the client size exceed $1k$. It indicates that the model performance of *LDP* mainly benefits from an environment with a large scale of distributed data.

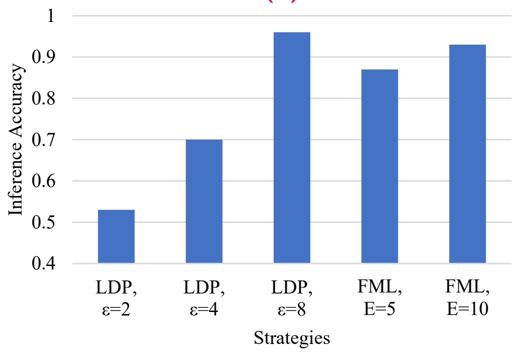
For *FML*, IID and non-IID setups were evaluated, that is, to distribute the data in a way where most labels evenly exist in each device or cluster in different devices. In both setups, misclassification rate decreases faster with more participants and stays at saturated level on 14% (IID) and 19%



(a) NYC Taxi



(b) BR2000



(c) Adult

Fig. 5.2: Privacy of different strategies (E: number of local passes in FML)

(non-IID), optimal in *BR2000* and *NYC Taxi* respectively when the number of clients reaches between 700 and 1000. *FML* can learn a useful model even if there are only a few clients at the early stage compared to *LDP*. It is obvious that the uneven distribution of data leads to a negative impact on *FML*, while *LDP* is free from the influence of data distribution since this strategy collects all data in the first place.

5.3.3 Privacy Loss

To understand the privacy loss of both strategies, inference accuracy is evaluated using general sample inference attacks. In this attack, we assume an adversary (e.g., untrusted aggregator) is able to decrypt the communication channel in both strategies and has basic knowledge about the types of the local training set (e.g., attribute type, candidate value). By observing the data transferred between a client and a server, i.e., perturbed data in *LDP* and model parameters in *FML*, the adversary can perform inference attack to determine which samples drawn from the same distribution belong to the client training set. A higher inference accuracy leads to greater privacy loss.

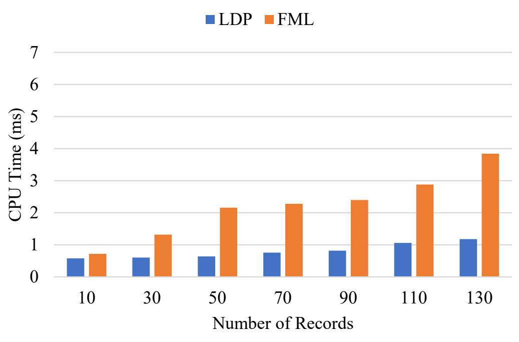
In *LDP*, the inference is performed by measuring the Manhattan distance between testing data and perturbed data, and a testing record is considered as a member of local dataset when its minimum distance is less than a threshold. As for *FML*, since adversary can obtain both global model and local updated one, by comparing the membership inference [78] on the two models, the local samples can be exposed. That is, given a threshold, if a record is recognized as a member in the inference on the local model but non-member on the global model, it is likely that this record belongs to the local set. We evaluated all settings on a testing dataset with half of the samples used in local training and the other half outside of the device, such that the random guess is 0.5. All results were reported under optimal threshold in their settings.

As shown in Fig. 5.2, *LDP* achieves a flexible control over privacy loss compared to *FML*. Except for the budget of 8, the inference accuracy is constrained to less than 80% and even 55% as privacy budget drops to 2 in all datasets. As for *FML*, the inference accuracy can reach over 80% among all datasets with 5 local passes and even 90% in *NYC Taxi* and *Adult* when local passes increase to 10. To improve communication efficiency, it commonly adds more computation to clients by iterating local updates multiple times before the aggregation step. This indicates that such fine-grained updates can significantly capture the details of local data and are vulnerable to malicious inference. In this case, *LDP* with low budget has stronger privacy guarantee than *FML* while the performance of classifier is the trade-off by revisiting model misclassification rate.

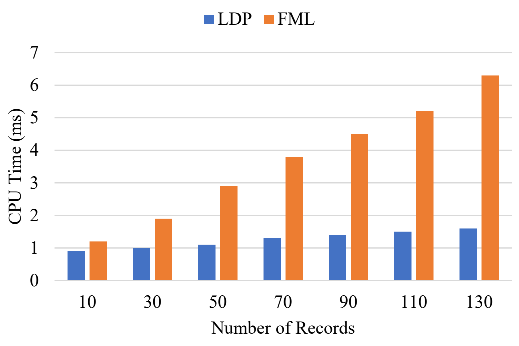
5.3.4 CPU Consumption

5.3.4.1 Client Side

The main client CPU consumption is on perturbation of data for *LDP* while *FML* spends most of the time updating the global model with local data. We review the CPU time against the average local dataset size of each device in Fig. 5.3. *FML* consumes more CPU to iterate through the data and grows linearly to over $3.8ms$ for *NYC Taxi* (resp. $6.3ms$ for *BR2000*) while *LDP* grows significantly slower and only reaches $1.2ms$ for *NYC Taxi* (resp. $1.7ms$ for *BR2000*). When the size of local dataset is small, the time will approximate preparation time such as parameters initialization since the real processing time is too short. The battery will drain faster under the setting of *FML*.

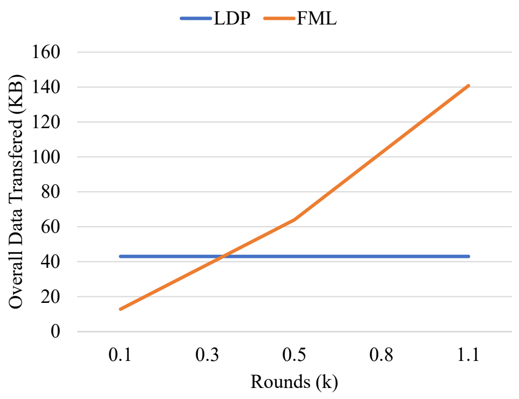


(a) NYC Taxi

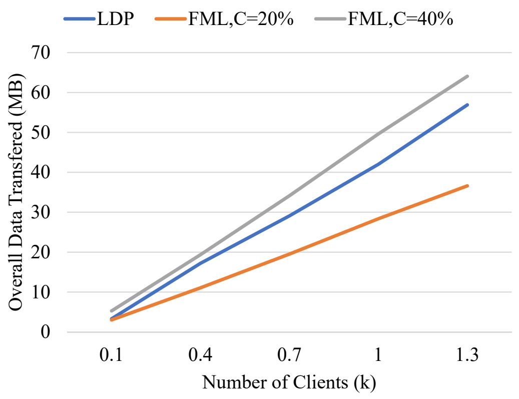


(b) BR2000

Fig. 5.3: Client CPU time



(a) Client bandwidth consumption, NYC Taxi



(b) Server bandwidth consumption, NYC Taxi

Fig. 5.4: Communication cost

5.3.4.2 Server Side

The computation resources of server are spent on pooling client's data and training the model in *LDP*. Obviously it consumes more CPU over server side to train the model compared to *FML* where the server only needs to coordinate clients and aggregates all received updates, since training workload is transferred to clients. For each model, 100 server training epochs take an average of $36s$ with 500 iterations in *LDP* while the aggregation and update process in *FML* take less than $1s$.

5.3.5 Communication Cost

5.3.5.1 Client Side

As for data transmission of client device, since *LDP* will collect all data, the transmission amount is constant to the size of local dataset while *FML* sends a number of parameters depending on model size. In Fig. 5.4a, *LDP* has a larger communication cost than *FML* when communication rounds are less than 300. Eventually, *FML* has 3x more cost due to frequent exchange of model updates with server.

5.3.5.2 Server Side

On the server side, since the communication cost against round change just aggregates all client's, we instead investigate the transferred data size against the number of clients by fixing round number at 1.1k (best model performance) for each client. As shown in Fig. 5.4b, *LDP* grows much faster with more participants than *FML* in communication cost since it is equivalent to collect the whole dataset combined from all clients. Due to the frequent interactions between clients and server, the accumulated transferred data grow quickly as well and reaches over 30MB when 20% of clients participate in each round and can outgrow *LDP* with 40% participation rate.

5.4 Discussion

Impact of Data and Training Procedure In this comparative study, we evaluated moderate type of data for generality. For "heavy data" like images and audio, we expect the trend of computation/network overhead will be similar to current comparative study but with widening gap. On the one hand, client CPU usage in FML will grow drastically as the model complexity also increases for such data while LDP remains the same. On the other hand, LDP will consume higher network usage given that perturbed data has a similar size of the original one. For LDP, we adopt the same straightforward training as FML for fair comparison. However, the model performance may be volatile to the privacy budget. Alternative training procedure using frequency-based statistics [2] can be adopted to improve the model quality and stability. The main idea is to generate synopsis such as histogram from perturbed data and synthesize training data from that synopsis.

Privacy Challenges in FML Even though *FML* provides a good property of intrinsic preservation of local data while delivering high-quality model, this strategy still faces many challenges on privacy protection and the reasons are three-fold. First, as shown in our empirical analysis, privacy control is limited for the submitted updates in *FML*, since the change of local pass number does not produce a significant influence over the privacy loss. Second, current *FML* heavily relies on encryption schemes to deliver secure aggregation and is susceptible to the inherited vulnerabilities of that designated encryption. Third, the system efficiency is liable to be degraded by the secure aggregation scheme, such as multi-party computation (MPC) [8] which is inherently computationally complex.

Unification of *FML* and *LDP*. Essentially, the aggregation step in *FML* is performing mean calculation on scattered data sources. Given that *LDP* has been frequently applied in such distributed analytical task [21], we can consider a unification approach that tackles the above challenges by integrating *FML* with *LDP*. The core idea is to inject ϵ-*LDP* perturbation to model updates before transmission. Specifically, on the client side, a set of training instructions are provided as usual to perform local training. In addition to batch size and the number of training passes, the client is also notified of *LDP* perturbation mechanism and a privacy budget ϵ. After parameter update W is derived, instead of submitting it immediately, the client will generate a noisy version $W + ldp(\epsilon)$. On the server side, noisy updates received from the selected clients will be merged to canceled the additive noises. This server-client interaction can repeat for multiple times with different budgets. If the perturbation is produced by a biased mechanism with non-zero mean, the server will further perform a calibration step on the aggregated result to obtain an accurate estimation.

In this way, the adversary can only recover noisy model updates even if the communication channel is intercepted. Besides, the level of perturbation can be flexibly negotiated on the fly. For example, if a participant finds the privacy budget unsatisfied, he/she can reject this round of training until the expectation is met. Furthermore, perturbation noise $ldp(\epsilon)$ is commonly generated with light computation, which can improve the overall efficiency compared to encryption scheme. Emerging works have tried to leverage such unification but the designs are still limited to particular genres of models[75]. In some aspects, the unification approach can always outperform the two originals given that the perturbation is presented in intermediate values and keep a high resolution of original data. Nonetheless, we leave their empirical study for future work.

5.5 Summary

We investigate two promising data analytic strategies for distributed setting while preserving user privacy. Both strategies are adopted in the same real machine learning problems and evaluated with extensive experiments under various system settings. The results show that local differential privacy mainly benefits from a large user population and consumes less CPU/battery on mobile devices while maintaining a rigorous privacy guarantee. Federated machine learning can adapt itself quickly for a moderate number of users and produce a learning model with higher quality while the fine-grained update is vulnerable to inference. Nonetheless, the data submitted with local differential privacy can be reused indefinitely for other tasks such as marginal release or itemset mining, while the model trained by FL is specified for one type of prediction task. As for future work, we plan to evaluate different unified solutions again each other using similar empirical framework. We also plan to propose new privacy-preserving method based on the comparative study.

Discussion and Conclusion

6

In this thesis, we investigated privacy invasion and protection in three levels of attack surface under the context of adversarial machine learning. We proposed a boundary differentially private layer against extraction attack, developed a learning-based eavesdropping system to infer indoor whereabouts and conducted a comparative study on two emerging privacy-preserving data analytics using local differential privacy and federated machine learning. As demonstrated in our study, unlike traditional privacy challenges in data management, securing AI systems against adversarial probing can be more difficult with the scrutiny of user requirements as well as the rapid evolvement of adversarial machine learning in different attack surfaces. This domain is in its infancy and calls for a broad contribution on various topics, such as balance of privacy and performance, personalized and on-demand design, rigorous privacy restriction in theoretical aspects. In particular, we would like to discuss three important directions that may shine the way toward privacy protection for the future.

Extraction and Defense on Sequential Model. As discussed in the literature review and our defense proposal, most existing extraction attacks focus on conventional classifiers where the output space is constrained to discrete and limited candidates. We are not aware of its influence on model adopted for sequential data, such as recurrent neural network (RNN), long short-term memory network (LSTM). Particularly, with the rise of voice assistant, automatic speech recognition has been widely applied in mobile phones, smart speakers and automatic driving, hence posing a compelling call to understand feasibility and damage. Compared to conventional model extraction, the amount of query budget is non-trivial given the complexity of sequential model and output combination, incurring high cost and system attention. Recently there has been a trend to provide offline service as Google rolls out on-device assistant and keyboard with the well-compressed model, which may alleviate this challenge and open up a new discussion.

Raw Data and Pervasive Obfuscation. Throughout this study, it is not difficult to see that the feasibility and severity of the attacks partly take advantage of accurate data produced in the pipeline of machine learning (e.g., faithful prediction, high-frequency sensor data, latest gradient). It would be interesting to ask whether such high-resolution values are necessary for all systems regardless of timing and input. A framework of pervasive obfuscation may be the potential solution to prevent privacy leakage in the first place and to significantly reduce adoption effort when new threats are encountered.

Sparse Data Collection. We have discussed the unification of local differential privacy and federated machine learning in the previous chapter. An implicit assumption in such unification is that there are enough participants to contribute to the learning process so that the obfuscation to the individual model can be sufficiently smoothed out. However, there are cases where this assumption will no hold, such as sparse data on particular illness where perturbation to all model updates are unrealistic to utility. One possible way to tackle this challenge is to consider the correlation between model updates or identify insignificant parameters using techniques from pruning study [13]. In this way, the privacy budget can be largely preserved for informative feedback and improve its utility when data are sparse.

To conclude, machine learning plays an increasingly important role in the rapidly transformed society and inevitably draws growing attention from adversaries. It is imperative to comprehend the source of privacy leakage in adversarial machine learning from a systematic view and never too late to prevent its immense impact. As Patrick Henry said in 1775, “Give me liberty, or give me death.”, we look forward to a world where true liberty not only lies in the freedom of expression but also in the choice to seclude the information about himself/herself.

References

[1]M. Abadi, A. Agarwal, P. Barham, and et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[2]M. Abadi, A. Chu, I. Goodfellow, *et al.*, “Deep learning with differential privacy,” in *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[3]C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional spaces,” in *ICDT*, J. V. den Bussche and V. Vianu, Eds., ser. Lecture Notes in Computer Science, vol. 1973, Springer, 2001, pp. 420–434.

[4]D. Angluin, “Queries and concept learning,” *Machine Learning*, vol. 2, pp. 319–342, 1987.

[5]Apple Privacy, *Our approach to privacy*, 2019. [Online]. Available: `https://www.apple.com/privacy/approach-to-privacy/`.

[6]D. Arp, E. Quiring, C. Wressnegger, and K. Rieck, “Privacy threats through ultrasonic side channels on mobile devices,” in *IEEE European Symp. Secur. Priva. (EuroS&P)*, 2017, pp. 35–47.

[7]K. Block and G. Noubir, “My magnetometer is telling you where i’ve been?: A mobile device permissionless location attack,” in *Proc. ACM Conf. Secur. Priva. Wireless Mobile Netw. (WiSec)*, 2018, pp. 260–270.

[8]K. Bonawitz, V. Ivanov, B. Kreuter, *et al.*, “Practical secure aggregation for privacy-preserving machine learning,” in *CCS*, 2017, pp. 1175–1191.

[9]L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10]L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth, 1984.

[11]R. F. Brena, J. García-Vázquez, C. E. Galván-Tejada, *et al.*, “Evolution of indoor positioning technologies: A survey,” *Journal of Sensors*, vol. 2017, 2630413:1–2630413:21, 2017.

[12]Q. A. Chen, Z. Qian, and Z. M. Mao, “Peeking into your app without actually seeing it: UI state inference and novel android attacks,” in *Proc. USENIX Secur. Symp.*, 2014, pp. 1037–1052.

[13]S. Chen, W. Wang, and S. J. Pan, “Deep neural network quantization via layer-wise optimization using limited training data,” in *The 33rd Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA*, AAAI Press, 2019, pp. 3329–3336.

[14]N. Chinchor, “MUC-4 evaluation metrics,” in *Proc. Conf. Message Understanding (MUC)*, 1992, pp. 22–29.

[15]K. Cho, W. Park, M. Hong, *et al.*, “Analysis of latency performance of bluetooth low energy (BLE) networks,” *Sensors*, vol. 15, no. 1, pp. 59–78, 2015.

[16]M. Christ, A. W. Kempa-Liehr, and M. Feindt, “Distributed and parallel time series feature extraction for industrial big data applications,” *arXiv preprint arXiv:1610.07717*, 2016.

[17]H. Cui, H. Zhang, G. R. Ganger, P. B. Gibbons, and E. P. Xing, “Geeps: Scalable deep learning on distributed gpus with a gpu-specialized parameter server,” in *EuroSys*, 2016.

[18]*Data protection in the eu*, Accessed: Nov. 19, 2018. [Online]. Available: `https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu`.

[19]P. Davidson and R. Piché, “A survey of selected indoor positioning methods for smartphones,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 2, pp. 1347–1370, 2017.

[20]J. Deng, W. Dong, R. Socher, *et al.*, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.

[21]R. Dewri, “Local differential perturbations: Location privacy under approximate knowledge attackers,” *IEEE Transactions on Mobile Computing*, vol. 12, no. 12, pp. 2360–2372, 2013.

[22]D. Dua and C. Graff, *UCI machine learning repository*, 2017. [Online]. Available: `http://archive.ics.uci.edu/ml`.

[23]J. C. Duchi, M. I. Jordan, and M. J. Wainwright, “Local privacy and statistical minimax rates,” in *IEEE Symposium on Foundations of Computer Science*, 2013, pp. 429–438.

[24]C. Dwork, “Differential privacy,” in *Automata, Languages and Programming, 33rd International Colloquium*, 2006, pp. 1–12.

[25]Ú. Erlingsson, V. Pihur, and A. Korolova, “Rappor: Randomized aggregatable privacy-preserving ordinal response,” in *CCS*, ACM, 2014, pp. 1054–1067.

[26]M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[27]X. Gao, B. Firner, S. Sugrim, *et al.*, “Elastic pathing: Your speed is enough to track you,” in *Proc. Int. Joint Conf. Pervas. and Ubiquitous Comput. (UbiComp)*, 2014, pp. 975–986.

[28]I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA*, 2015.

[29]J. Goslinski, M. Nowicki, and P. Skrzypczynski, “Performance comparison of ekf-based algorithms for orientation estimation on android platform,” *IEEE Sensors Journal*, vol. 15, no. 7, pp. 3781–3792, 2015.

[30]B. Gozick, K. P. Subbu, R. Dantu, and T. Maeshiro, “Magnetic maps for indoor navigation,” *IEEE Trans. on Instrumentation and Measurement*, vol. 60, no. 12, pp. 3883–3891, 2011.

[31]L. Grustniy, *What’s wrong with ’legal’ commercial spyware.* 2018. [Online]. Available: `https://www.kaspersky.com/blog/stalkerware-spouseware/26292/`.

[32]T. Guardian, *Amazon’s alexa recorded private conversation and sent it to random contact*, Accessed: July 17, 2018. [Online]. Available: `https://www.theguardian.com/technology/2018/may/24/amazon-alexa-recorded-conversation`.

[33]J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang, “Accomplice: Location inference using accelerometers on smartphones,” in *Int. Conf. Commun. Syst. Netw. (COMSNETS)*, 2012, pp. 1–9.

[34]D. M. Harris and S. L. Harris, “Digital design and computer architecture,” 2007.

[35]S. Hemminki, P. Nurmi, and S. Tarkoma, “Accelerometer-based transportation mode detection on smartphones,” in *Proc. ACM Conf. Embedded Netw. Sens. Syst. (SenSys)*, 2013, 13:1–13:14.

[36]J. Hua, Z. Shen, and S. Zhong, “We can track you if you take the metro: Tracking metro riders using accelerometers on smartphones,” *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 2, pp. 286–297, 2017.

[37]*Indooratlas*, Accessed: Nov 20, 2018. [Online]. Available: `http://www.indooratlas.com/`.

[38]IPUMS, *Harmonized international census data for social science and health research*, 2018. [Online]. Available: `https://international.ipums.org/international/index.shtml`.

[39]J. Isaak and M. J. Hanna, “User data privacy: Facebook, cambridge analytica, and privacy protection,” *Computer*, vol. 51, no. 8, pp. 56–59, 2018.

[40]M. Jagielski, A. Oprea, B. Biggio, *et al.*, “Manipulating machine learning: Poisoning attacks and countermeasures for regression learning,” in *2018 IEEE Symposium on Security and Privacy, SP 2018, San Francisco, California, USA*, pp. 19–35.

[41]M. Juuti, S. Szyller, A. Dmitrenko, S. Marchal, and N. Asokan, “Prada: Protecting against dnn model stealing attacks,” *CoRR*, vol. abs/1805.02628, 2018.

[42]P. Kairouz, K. Bonawitz, and D. Ramage, “Discrete distribution estimation under local privacy,” in *Proceedings of the 33nd International Conference on Machine Learning, ICML*, vol. 48, 2016, pp. 2436–2444.

[43]P. Kairouz, S. Oh, and P. Viswanath, “Extremal mechanisms for local differential privacy,” *Journal of Machine Learning Research*, vol. 17, 17:1–17:51, 2016.

[44]W. Kang and Y. Han, “Smartpdr: Smartphone-based pedestrian dead reckoning for indoor localization,” *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2906–2916, 2015.

[45]E. Keogh, S. Chu, D. Hart, and M. Pazzani, “An online algorithm for segmenting time series,” in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 2001, pp. 289–296.

[46]M. Kesarwani, B. Mukhoty, V. Arya, and S. Mehta, “Model extraction warning in mlaas paradigm,” in *Annual Computer Security Applications Conference*, 2018.

[47]A. Khalajmehrabadi, N. Gatsis, and D. Akopian, “Modern wlan fingerprinting indoor positioning methods and deployment challenges,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1974–2002, 2017.

[48]A. Krizhevsky, *The cifar-10 dataset*. [Online]. Available: `https://www.cs.toronto.edu/~kriz/cifar.html`.

[49]D. F. Kune, J. Kölndorfer, N. Hopper, and Y. Kim, “Location leaks over the GSM air interface,” in *Proc. Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2012.

[50]Y. LeCun, *The mnist database of handwritten digits*. [Online]. Available: `http://yann.lecun.com/exdb/mnist/`.

[51]J. Lee and D. Kifer, “Concentrated differentially private gradient descent with adaptive per-iteration privacy budget,” in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2018.

[52]T. Lee, B. Edwards, I. Molloy, and D. Su, “Defending against model stealing attacks using deceptive perturbations,” *CoRR*, vol. abs/1806.00054, 2018.

[53]B. Li, T. Gallagher, A. G. Dempster, and C. Rizos, “How feasible is the use of magnetic field alone for indoor positioning?” In *Int. Conf. Indoor Positioning and Indoor Navigation (IPIN)*, 2012, pp. 1–9.

[54]H. Li, H. Zhu, S. Du, X. Liang, and X. Shen, “Privacy leakage of location sharing in mobile social networks: Attacks and defense,” *IEEE Trans. Dependable and Security Comput.*, vol. 15, no. 4, pp. 646–660, 2018.

[55]M. Li, Y. Meng, J. Liu, *et al.*, “When csi meets public wifi: Inferring your mobile phone password via wifi signals,” in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2016, pp. 1068–1079.

[56]N. Li, W. Qardaji, and D. Su, “On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy,” in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS ’12, Seoul, Korea, 2012, 32–33.

[57]W. Y. B. Lim, N. C. Luong, D. T. Hoang, *et al.*, “Federated learning in mobile edge networks: A comprehensive survey,” *IEEE Communications Surveys Tutorials*, pp. 1–1, 2020.

[58]D. Lowd and C. Meek, “Adversarial learning,” in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ser. KDD ’05, ACM, 2005, pp. 641–647.

[59]H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *AISTATS*, 2017.

[60]M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, “Touchsignatures: Identification of user touch actions and pins based on mobile sensor data via javascript,” *Journal of Info. Secur. and Appl.*, vol. 26, pp. 23 –38, 2016.

[61]M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, “Touchsignatures: Identification of user touch actions based on mobile sensors via javascript,” in *Proc. ACM ASIA Conf. Comput. Commun. Secur. (AsiaCCS)*, 2015, p. 673.

[62]Y. Michalevsky, D. Boneh, and G. Nakibly, “Gyrophone: Recognizing speech from gyroscope signals,” in *Proc. USENIX Secur. Symp.*, 2014, pp. 1053–1067.

[63]Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, “Powerspy: Location tracking using mobile device power analysis,” in *Proc. USENIX Secur. Symp.*, 2015, pp. 785–800.

[64]A. Mosenia, X. Dai, P. Mittal, and N. K. Jha, “Pinme: Tracking a smartphone user around the world,” *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 3, pp. 420–435, 2018.

[65]Y. Murata, K. Kaji, K. Hiroi, and N. Kawaguchi, “Pedestrian dead reckoning based on human activity sensing knowledge,” in *Proc. Int. Joint Conf. Pervas. and Ubiquitous Comput. (UbiComp)*, 2014, pp. 797–806.

[66]S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir, “Inferring user routes and locations using zero-permission mobile sensors,” in *IEEE Symp. Secur. and Priva. (S&P)*, 2016, pp. 397–413.

[67]A. Narayanan and V. Shmatikov, “Robust de-anonymization of large sparse datasets,” in *S&P*, 2008, pp. 111–125.

[68]K. O’Flaherty, *Whatsapp users targeted by spyware – here’s what you need to know.* 2019. [Online]. Available: `https://www.forbes.com/sites/kateoflahertyuk/2019/05/14/whatsapp-users-targeted-with-israeli-spyware-heres-what-you-need-to-know/`.

[69]S. J. Oh, M. Augustin, B. Schiele, and M. Fritz, “Towards reverse-engineering black-box neural networks,” in *International Conference on Learning Representations*, 2018.

[70]A. Ometov, A. Levina, P. Borisenko, *et al.*, “Mobile social networking under side-channel attacks: Practical security challenges,” *IEEE Access*, vol. 5, pp. 2591–2601, 2017.

[71]T. Orekondy, B. Schiele, and M. Fritz, “Knockoff nets: Stealing functionality of black-box models,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2019.

[72]E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, “Accessory: Password inference using accelerometers on smartphones,” in *ACM Workshop on Mobile Comput. Syst. and Appl. (HotMobile)*, 2012, 9:1–9:6.

[73]S. Pal, Y. Gupta, A. Shukla, *et al.*, “Activethief: Model extraction using active learning and unannotated public data,” in *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[74]N. Papernot, P. McDaniel, I. Goodfellow, *et al.*, “Practical black-box attacks against machine learning,” in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, Abu Dhabi, United Arab Emirates, 2017, pp. 506–519.

[75]V. Pihur, A. Korolova, F. Liu, *et al.*, “Differentially-private "draw and discard" machine learning,” *CoRR*, vol. abs/1807.04369, 2018.

[76]*Privacy & data security update*, Accessed: Nov. 16, 2018. [Online]. Available: `http://www.ftc.gov/reports/` (visited on May 16, 2018).

[77]E. Quiring, D. Arp, and K. Rieck, “Forgotten siblings: Unifying attacks on machine learning and digital watermarking,” *IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 488–502, 2018.

[78]A. Salem, Y. Zhang, M. Humbert, *et al.*, “Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models,” in *NDSS*, 2019.

[79]M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, IEEE Computer Society, 2018, pp. 4510–4520.

[80]J.-l. Shen, J.-w. Hung, and L.-s. Lee, “Robust entropy-based endpoint detection for speech recognition in noisy environments,” in *Proc. Int. Conf. on Spoken Language Processing*, 1998.

[81]R. Shokri, M. Stronati, and V. Shmatikov, “Membership inference attacks against machine learning models,” *IEEE Symposium on Security and Privacy*, pp. 3–18, 2017.

[82]T. W. Smith, P. Marsden, M. Hout, and J. Kim, *General social survey*, 2013. [Online]. Available: `https://gss.norc.org/Get-The-Data`.

[83]H. Soroush, K. Sung, E. G. Learned-Miller, B. N. Levine, and M. Liberatore, “Turning off GPS is not enough: Cellular location leaks over the internet,” in *Proc. Priva. Enhancing Technol. Symp. (PETs)*, 2013, pp. 103–122.

[84]K. M. Ting, F. T. Liu, and Z. Zhou, “Isolation forest,” in *IEEE Int. Conf. Data Mining (ICDM)*, vol. 00, 2008, pp. 413–422.

[85]N. TLC, *Trip record data*, 2016. [Online]. Available: `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`.

[86]F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, “Stealing machine learning models via prediction apis,” in *Proceedings of the 25th USENIX Conference on Security Symposium*, 2016, pp. 601–618.

[87]L. G. Valiant, “A theory of the learnable,” in *ACM Symposium on Theory of Computing*, 1984.

[88]J. Verble, “The NSA and edward snowden: Surveillance in the 21st century,” *SIGCAS Comput. Soc.*, vol. 44, no. 3, pp. 14–20, 2014.

[89]B. Wang and N. Z. Gong, “Stealing hyperparameters in machine learning,” in *IEEE Symposium on Security and Privacy*, 2018, pp. 36–52.

[90]N. Wang, X. Xiao, Y. Yang, *et al.*, “Collecting and analyzing multidimensional data with local differential privacy,” in *ICDE*, 2019, pp. 638–649.

[91]S. L. Warner, “Randomized response: A survey technique for eliminating evasive answer bias,” *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[92]S. L. Warner, “Randomized response: A survey technique for eliminating evasive answer bias,” *Journal of the American Statistical Association*, vol. 60(309), pp. 63–6, 1965.

[93] T. Watanabe, M. Akiyama, and T. Mori, “Routedetector: Sensor-based positioning system that exploits spatio-temporal regularity of human mobility,” in *USENIX Workshop on Offensive Technol. (WOOT)*, 2015.

[94] S. S. Wilks, “The large-sample distribution of the likelihood ratio for testing composite hypotheses,” *The Annals of Mathematical Statistics*, vol. 9, no. 1, pp. 60–62, 1938.

[95] M. Won, A. Mishra, and S. H. Son, “Hybridbaro: Mining driving routes using barometer sensor of smartphone,” *IEEE Sensors Journal*, vol. 17, no. 19, pp. 6397–6408, 2017.

[96] W. Xu, Y. Qi, and D. Evans, “Automatically evading classifiers: A case study on PDF malware classifiers,” in *Annual Network and Distributed System Security Symposium*, 2016.

[97] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, 12:1–12:19, 2019.

[98] H. Yu, K. Yang, T. Zhang, *et al.*, “Cloudleak: Large-scale deep learning models stealing through adversarial examples,” in *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*, 2020.

[99] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, “Calm: Consistent adaptive local marginal for marginal release under local differential privacy,” in *CCS*, 2018, pp. 212–229.

[100] L. Zhou, S. Du, H. Zhu, *et al.*, “Location privacy in usage-based automotive insurance: Attacks and countermeasures,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 196–211, 2019.

[101] X. Zhou, S. Demetriou, D. He, *et al.*, “Identity, location, disease and more: Inferring your secrets from android public resources,” in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 1017–1028.